spielen, lernen, arbeiten mit dem ti-99/4A

K.-J. Schmidt
G.-P. Raabe



spielen, lernen, arbeiten
mit dem *ti-99/4A*

spielen, lernen, arbeiten mit dem ti-99/4A

Georg-Peter Raabe
Klaus-Jürgen Schmidt



BERKELEY · PARIS · DÜSSELDORF

Anmerkung:
TI-99/4A ist ein eingetragenes Warenzeichen von Texas Instruments Inc., Houston/Texas

Umschlaggestaltung und Satz: tgr – typo-grafik-repro gmbh., remscheid
Gesamtherstellung: Druckerei Hub. Hoch, Düsseldorf

ISBN 3-88745-039-6
1. Auflage 1984

Inhaltsverzeichnis

Vorwort

Dieses Buch vermittelt Ihnen theoretische und praktische Kenntnisse für die Anwendung des Heimcomputers TI-99/4A. Die große Vielfalt der Einsatzmöglichkeiten eines solchen Computers kann besonders für den Anfänger wegen der Unüberschaubarkeit des Gebietes leicht zu einer gewissen Hilflosigkeit führen. Als gar nicht seltene Folge führt dies schließlich dazu, daß dieses zwar preiswerte, aber doch nicht ganz billige Gerät nach kurzer Zeit nicht mehr benutzt wird.

Damit dies nicht geschieht, sollten Sie Ihren Rechner von Anfang an, schrittweise aufbauend, immer besser kennen- und beherrschen lernen. Die Absicht dieses Buches ist es, Ihnen einen Weg für die Eroberung Ihres TI-99/4A zu zeigen, der in angenehmer, spielerischer Art verläuft und Ihnen zugleich angemessenes, solides EDV-Handwerk vermittelt.

In diesem Sinne ist das Ziel dieses Buches formuliert. Über die drei Phasen SPIELEN, LERNEN und ARBEITEN sollen dem Leser Fähigkeiten vermittelt werden, die er bei der Programmierung und Anwendung seines TI-99/4A direkt benötigt und die ihn in die Lage versetzen, auch größere Programmkomplexe anderer Programmierer und Hersteller zu verstehen und für seinen eigenen Bereich zu ändern und anzuwenden.

Wir wünschen Ihnen hierbei viel Spaß und Erfolg.

Passau, im November 1983

Georg-Peter Raabe
Klaus Jürgen Schmidt

Kapitel 1
Einführung

Computer sind aus dem heutigen Leben nicht mehr wegzudenken. Wir begegnen ihnen mehr oder weniger unmittelbar in vielen Bereichen des täglichen Lebens: als Mikroprozessoren in Haushaltsgeräten oder als Bordcomputer im Auto. Als Personal Computer und Arbeitsplatzrechner dringen sie in den Bürobereich vor. In Labors und in der Montage haben sie seit einiger Zeit einen festen Platz erobert. In der Form von Großrechnern und Jumbocomputern werden die meisten nur mittelbare Erfahrung mit diesen Rechnern haben: Rechnungen, Versicherungspolicen, Bescheide von Finanzämtern und Behörden werden in erheblichem Umfang unter Verwendung von Rechenanlagen erstellt. Hinter der Wetterkarte und der Wettervorhersage, die uns das Fernsehen täglich liefern, steckt der Einsatz von Größtrechenanlagen, die über 100 Millionen komplexe Rechenoperationen pro Sekunde ausführen.

Die Einsatzmöglichkeiten von Computern sind inzwischen von so großer Vielfalt, daß es unmöglich ist, sie auch nur annäherungsweise zu erwähnen, selbst bei Beschränkung auf die Rechner am unteren Ende des Leistungsspektrums. In Anbetracht der ungeheuren Vielfalt haben wir ganz bewußt eine Auswahl getroffen, die mit einer gewissen Grundausstattung auskommt. Dieses Buch, *Spielen – Lernen – Arbeiten mit dem TI-99/4A,* möchte Ihnen einen spielerisch angenehmen Weg in die Benutzung Ihres Computers aufzeigen und Sie dazu anregen, die gezeigten Programme und Anwendungen weiterzuentwickeln und auf Neues auszudehnen.
Die Programme laufen alle auf der weiter unten beschriebenen Basisausstattung.
Da unser Buch gerade für Anfänger in der „Computerei" eine Hilfe sein soll und sich die meisten mit ihrer Rechnerausstattung erst schrittweise an aufwendigere und damit auch teurere Geräte vorarbeiten, werden Programme, die nicht ohne Disketten oder Drucker auskommen, nicht behandelt. Wir sind der Überzeugung, daß jemand, der sich in dem hier

abgesteckten Umfang mit seinem Rechner vertraut gemacht hat, ohne weiteres ähnliche Anwendungen mit Disketten, Druckern oder Datenübertragungseinrichtungen unter Zuhilfenahme der zugehörigen technischen Unterlagen selbständig in Angriff nehmen und bewältigen kann.

In den folgenden Abschnitten dieses Kapitels werden das TI-99/4A-System beschrieben sowie die Möglichkeiten der Erweiterung und Ergänzung der Hardware und der Software aufgezeigt. Da laufend neue Komponenten angeboten werden, kann diese Aufzählung nur einen momentanen Abriß geben. Weiter geben wir Ihnen einige Tips und Bedienungshinweise für die Benutzung Ihres TI-99/4A, die uns beim Schreiben und Austesten der Programme als nützlich erschienen sind.

BESCHREIBUNG DES TI-99/4A

Wir gehen davon aus, daß Sie die Beschreibungen und Unterlagen, die Sie mit Ihrem Rechner erhielten, sorgfältig studiert haben und über die Komponenten ausreichend Bescheid wissen, so daß eine Zusammenfassung des Wesentlichen hier ausreicht. In dem Kapitel „Lernen" stellen wir Ihnen einige Einzelheiten aus dem „Innenleben" Ihres TI-99/4A vor.

Ein in sich funktionsfähiges Minimalsystem besteht aus:

Konsole,
Netzteil,
PAL-Modulator,
den dazugehörenden Verbindungskabeln,
der Bedienungsanleitung
und einem Fernsehgerät.

Zusammenbau und Inbetriebnahme sind in den Bedienungsanleitungen beschrieben.

Die Konsole enthält den eigentlichen Rechner, den 16-Bit-Mikroprozessor TMS 9900 von Texas Instruments, und den Arbeitsspeicher mit 16 KB RAM. Außerdem sind in der Konsole in Permanentspeichern, die nur gelesen werden können (ROMs), das Betriebssystem und der Interpreter für die Programmiersprache BASIC (TI-BASIC) untergebracht sowie Schnittstellen für den Anschluß weiterer Geräte. Die Konsole bietet mit einer Schreibmaschinentastatur einen bei Heimcomputern nicht immer üblichen Bedienungskomfort.

Für die Versorgung Ihres Computers mit der richtigen Spannung sorgt das mitgelieferte Netzteil. Ohne ausreichende Kenntnisse in moderner Digitalelektronik sollten Sie an der Stromversorgung keine Reparaturen oder gar Experimente durchführen. Wenn Sie dabei Ihren Rechner „schlachten", ist das unter Umständen noch das wenigste, was Ihnen bei unfachmännischem Umgang mit der Stromversorgung passieren kann.

Der PAL-Modulator setzt die Informationen aus dem Rechner in ein Fernsehsignal um, das mit einem handelsüblichen Fernsehgerät auf Kanal 36 empfangen werden kann. PAL ist das in Deutschland und einer Reihe weiterer Länder benutzte Verfahren für die Übermittlung von farbigen Fernsehbildern. Genauso wie Sie die von der Post ausgestrahlten Farbfernsehsendungen auch mit einem einfachen Schwarzweiß-Fernsehgerät empfangen können, können Sie statt des Farbfernsehers an Ihren TI-99/4A auch einen Schwarzweiß-Fernsehapparat anschließen. Anstelle der Farben erhalten Sie dann natürlich nur die entsprechenden Graustufen.

Mit dieser Minimalkonfiguration können die meisten Programme nachvollzogen werden. Die Basisausstattung, auf der die Programme in diesem Buch entwickelt und ausgetestet wurden, enthält noch folgende Erweiterungen, die nach unserer Erfahrung erst ein angemessenes Arbeiten mit dem TI-99/4A ermöglichen:

Kassettenrecorder,
Kassettenrecorderkabel,
Extended BASIC Modul und
Programmierhandbuch TI-BASIC /Extended BASIC.

Wenn Sie einen geeigneten Kassettenrecorder haben (am besten einen mit Fernbedienung), dann kosten diese Ergänzungen zusammen rund DM 370,– (Kabel ca. DM 30,–, Extended BASIC Modul ca. DM 300,– und das Programmierhandbuch ca. DM 40,–). Diese Angaben sind wegen der zum Teil erheblichen Preisbewegungen auf diesem Markt nur ungefähr und sollen Ihnen lediglich eine grobe Abschätzung ermöglichen.

Der Kassettenrecorder ist ab einer gewissen Programmierstufe unbedingt erforderlich, da Sie dann Programme langfristig speichern können. Sie müssen sonst jedesmal die Programme neu eingeben, was recht bald frustrierend werden kann. Darüber hinaus ist es für das Experimentieren mit Programmen wichtig, Varianten auszuprobieren, was bei

entsprechend großen Programmen praktisch nur mit einem externen Speichermedium wie einem Kassettenrecorder praktikabel ist. Selbstverständlich wäre ein Diskettensystem dazu noch besser geeignet. Es ist letztlich eine Frage der einsetzbaren finanziellen Mittel. Trotz gewisser Einschränkungen, die der Kassettenbetrieb im Vergleich zu einem Diskettensystem notwendigerweise bedingt, ist dies letztlich auch aus Preisgründen gerade für den Anfang das geeignete externe Speichermedium. Sehr zu empfehlen ist an dieser Stelle das spezielle Kassettenlaufwerk von Texas Instruments (für ca. DM 180,– einschließlich Kabel), das in seinen technischen Details für den Einsatz mit dem TI-99/4A angepaßt ist und somit auch langfristig eine entsprechende Qualität sicherstellt. Aber für den Anfang reicht, wie oben erwähnt, auch ein normaler Recorder, wie es die Autoren über einige Zeit hin ausprobiert haben. Die richtige Einstellung von Lautstärke und Tonblende an Ihrem Recorder müssen Sie wohl oder übel durch einiges Herumprobieren ermitteln.

Mit der Programmiersprache Extended BASIC, also einem gegenüber dem normalerweise vorhandenen TI-BASIC in seinem Sprachumfang erweiterten BASIC, verfügen Sie über eine Sprache, die auf dieser Ebene die optimale Nutzung Ihres Rechners ermöglicht. Mit dem Modul wird meist nur ein englisches Handbuch geliefert. Wir empfehlen daher das Programmierhandbuch TI-BASIC / Extended BASIC für Anfänger und Fortgeschrittene von Texas Instruments, Learning Center (ISBN 3-88078-039-0), das sowohl das standardmäßig vorhandene TI-BASIC wie auch das Extended BASIC behandelt. In einem ersten Teil werden Sie schrittweise in die Programmierung in BASIC eingeführt, ein weiterer Teil des Buches enthält einen alphabetisch angeordneten Nachschlageteil aller BASIC-Anweisungen und -Subroutinen sowie im Anhang einen übersichtlichen Tabellenteil. Es sei an dieser Stelle ausdrücklich auf das oben erwähnte Buch verwiesen, besonders auf die Befehlsliste im Abschnitt C mit ausführlichen Erklärungen und Beispielen.

ERGÄNZUNGEN UND ERWEITERUNGEN DES TI-99/4A-SYSTEMS

Der TI-99/4A ist der zentrale Teil eines ausbaufähigen Systems sowohl in der Hardware als auch in der Software. Mit dem Begriff Hardware bezeichnet man alle physikalisch vorhandenen Teile eines Rechnersy-

stems, also alle Geräte. Die Software ist dagegen die Menge der auf einem Rechner laufenden Programme. Es sind also die Teile, die man nicht anfassen kann. Man unterscheidet bei der Software meistens nach Betriebssoftware und Anwendungssoftware. Die Betriebssoftware sorgt für die allgemeine Benutzbarkeit eines Rechnersystems, sie stellt jedoch noch keine direkte Lösung von Aufgaben dar, sie hilft lediglich dabei. Zur Betriebssoftware gehören das eigentliche Betriebssystem und die Programme, die die Programmiersprachen in eine dem Rechner verständliche Form umsetzen. Diese Programme heißen Compiler, wenn sie das in einer bestimmten Sprache geschriebene Programm (Source-Code oder Quellenprogramm genannt) erst komplett in die rechnerinterne Sprache übersetzen und dann in einem davon unabhängigen Schritt dieses sogenannte Maschinenprogramm (auch Object-Code genannt) zur Ausführung kommt. Eine andere Form sind die Interpreter, die das in der höheren Programmiersprache geschriebene Programm schrittweise lesen und interpretieren und diese Einzelschritte sofort ausführen. TI-BASIC und Extended BASIC sind beides Sprachen, die von Interpretern bearbeitet werden. Die Interpreter-Programme dafür sind in ROMs (Read Only Memory) innerhalb des Konsolgehäuses bzw. im Solid State Modul untergebracht.

Die Lösung von konkreten Aufgaben erfolgt mit der sogenannten Anwendungssoftware. Diese Programme müssen für jedes zu lösende Problem entsprechend erstellt werden, entweder von Ihnen selbst oder von anderen, die dann dafür eine entsprechende Bezahlung erwarten. Die Programmierung großer oder komplexer Programme ist eine sehr anspruchsvolle Arbeit, die im allgemeinen eine angemessene Ausbildung und Erfahrung erfordert, also entsprechend hoch qualifiziertes Personal. Der vollständige Arbeitsvorgang von der Erarbeitung einer Lösung, über die Programmierung, das Austesten und die Dokumentation mit Erstellung der Bedienungsanleitungen ist darüber hinaus ein recht langwieriges Unternehmen. Wir erwähnen dies hier, um Ihnen zumindest andeutungsweise ein Gefühl dafür zu geben, warum Software oft so teuer erscheint. Um es etwas konkreter zu machen: Für die Entwicklung eines brauchbaren Buchhaltungssystems muß man Kosten von DM 50.000,– aufwärts veranschlagen. Daß Sie andererseits einen Interpreter für Extended BASIC, ein Programm, das eher komplexer als ein Buchhaltungsprogramm ist, für rund DM 300,– bekommen können, ist nur dadurch möglich, daß entsprechende Stückzahlen verkauft werden.

Über die oben beschriebene Basiskonfiguration hinaus sind derzeit (September 1983) folgende Erweiterungen erhältlich:

Hardware

Fernbedienung: Zur Steuerung von Objekten (Zeichen, Figuren, Sprites) auf dem Bildschirm mittels eines Steuerhebels.

Sprach-Synthesizer: Auf rein elektronischer Basis, also ohne vorbesprochene Bänder, kann mittels BASIC-Anweisungen Sprache ausgegeben werden. Das System verfügt über einen Wortschatz von 400 englischen Wörtern. Die Möglichkeit, eigene Wörter in anderen Sprachen wie z.B. Deutsch zu generieren, soll mit einem Speech Editor angeboten werden.

Peripherie-Erweiterungssystem-Box: Für den Ausbau des Systems mit anspruchsvoller Peripherie ist diese Erweiterungsbox erforderlich. Sie enthält ein ausreichend leistungsstarkes Netzteil für die Stromversorgung der Karten und der anschließbaren Geräte. In die Box können Karten mit der Steuerelektronik für weitere periphere Geräte eingesteckt werden.

Diskettensystem: Das Diskettensystem besteht aus einer Steuerkarte, die in die Erweiterungsbox gesteckt wird, und den Diskettenlaufwerken. Diese Steuerung kann bis zu drei Floppy-Disk-Laufwerke betreiben. Eine Floppy Disk ist eine Scheibe von beispielsweise 5¼ Zoll Durchmesser aus Plastikmaterial und mit einer magnetisierbaren Schicht versehen. Durch Magnetisierung dieser Schicht wird Information auf die Diskette geschrieben und durch Abtasten der Magnetisierung wieder gelesen. Ein Diskettenlaufwerk kann direkt in die Erweiterungsbox eingebaut werden. Mit einem separat aufgestellten Doppellaufwerk ist der maximale Ausbau mit drei Laufwerken erreicht. Die technischen Daten der Disketten: single side, single density, 89 KB, 463 msec durchschnittliche Zugriffszeit.

RS 232-Schnittstelle: Mit Hilfe dieser Karte können Daten vom TI-99/4A zu anderen Geräten oder Rechnern übertragen werden. Mit RS 232 wird ein international genormtes Verfahren für die Datenübertragung bezeichnet. Auf der Karte sind zwei serielle Anschlußstellen und eine parallele Anschlußstelle (Centronics Port) vorhanden. Es sind Übertragungsraten von 110, 300, 600, 1200, 2400, 4800 oder 9600 Bits pro Sekunde möglich. Über diese Schnittstelle ist auch der Anschluß von

Modems möglich, wodurch die Datenübermittlung zu praktisch beliebig weit entfernten Geräten oder Rechnern ermöglicht wird.

Matrixdrucker: Damit können sowohl Texte als auch Graphiken ausgegeben werden. Der Anschluß erfolgt über die RS-232-Schnittstellenkarte.

Solid State Software Programm-Module: Diese steckbaren Module enthalten in ROMs (Read Only Memory) sofort ablauffähige Programme. Ein großer Teil der nachfolgend erwähnten Programme wird als Solid State Modul geliefert.

Speichererweiterungs-Karte: Diese Karte erweitert die frei verfügbare Speicherkapazität um 32 KBytes auf insgesamt 48 KBytes.

Sprachen

Extended BASIC: Solid State Modul mit erweitertem BASIC. Die in diesem Buch beschriebenen Programme wurden überwiegend in Extended BASIC geschrieben und benutzen im allgemeinen auch die darin enthaltenen Erweiterungen.

TI-Logo: Eine in den USA speziell für Kinder entwickelte Programmiersrache. Als zusätzliche Ausstattung ist die Speichererweiterung auf 48 KB erforderlich. Ein Kassettenrecorder oder besser noch ein Diskettensystem ist sehr zu empfehlen.

UCSD-Pascal: Pascal ist eine hochstrukturierte, sehr effiziente Programmiersprache. Benötigt werden dafür die Speichererweiterung auf 48 KB, das Diskettensystem und die P-Code-Karte.

Editor/Assembler: Damit können Sie direkt den Mikroprozessor TMS 9900 in Assembler programmieren, wodurch die Eigenschaften des Prozessors optimal genutzt werden können und Programme mit besonders schneller Ausführung geschrieben werden können. In Assembler geschriebene Programme sind allein ausführbar oder können auch in BASIC-Programme eingebunden werden. Erforderlich sind dafür die Speichererweiterung um 32 KB und das Diskettensystem.

Mini Memory: Dieses Solid State Modul erweitert unter anderem die frei verfügbare Speicherkapazität um 4 KB. Ferner ist damit die Benutzung des Assemblers möglich (eine genaue Beschreibung enthält das Buch von Rainer Bernert: TMS 9900 Assembler Handbuch für das Mini Memory. 1. Auflage, Wien 1983. Texas Instruments Wien).

Anwendungssoftware

Der Katalog der Anwendungssoftware enthält Programme aus den unterschiedlichsten Gebieten. Eine vollständige Aufzählung ist hier nicht möglich, zum einen wegen des Umfanges und zum anderen aus Gründen der Aktualität. Erwähnt seien deshalb nur einige mehr oder weniger willkürlich herausgegriffene Beispiele:

Lernprogramme: Rechenübungen in Form von unterhaltsamen Spielen; Music Maker, ein Programm zum Komponieren von kleinen Musikstücken.

Unterhaltung: Eine große Palette von elektronischen Unterhaltungsspielen wie Adventure, Voodoo Castle, Geisterstadt, Alpinist, Autorennen, Fitness-Trainer, Munch Man, Othello, Invaders, Yahtzee und viele andere mehr.

Organisation: Programme für Haushaltsplanung, Lagerverwaltung, Rechnungsstellung, Versandlisten, Text- und Dateiverwaltung, Datenverwaltung und -analyse, Statistik, Buchungsjournal etc. Ein Teil dieser Programme setzt ein entsprechend ausgebautes System voraus.

Sonstige Anwendungsgebiete: Elektrotechnik, Mathematik, Baustatik, Simplex-Verfahren, Betriebs- und Finanzwesen, Baurentabilität und -finanzierung, Kalkulation und Zuschnitt für Fenster etc.

Eine aktuelle Liste der verfügbaren Software können Sie von Texas Instruments oder Ihrem Händler erhalten. Neben der Software, die von TI vertrieben wird, gibt es eine ganze Reihe von Programmen, die von anderen Firmen oder unabhängigen Programmautoren direkt angeboten werden. Weitere Informationen darüber können Sie über die verschiedenen in der Bundesrepublik bestehenden Anwender-Clubs von TI-Home-Computern erfahren.

Diese Übersicht zeigt, daß Sie mit dem TI-99/4A über ein System verfügen, dessen Hardware umfangreiche Erweiterungsmöglichkeiten bietet und zu dem es eine respektable Palette fertiger Software gibt. Dieses Buch beabsichtigt jedoch nicht, Ihnen ein Führer für den Erwerb von Software zu sein, sondern Ihnen Fähigkeiten für die selbständige Programmierung zu vermitteln. Diese Kenntnisse und Fähigkeiten werden Sie auch viel besser in die Lage versetzen, sich selbst ein Urteil über das bestehende Software-Angebot zu machen.

BEDIENUNGSHINWEISE

Editieren

Es wird Ihnen mit Sicherheit auch wie allen anderen ergehen, daß die geschriebenen Programme nicht auf Anhieb fehlerfrei sind, meist auch nicht frei von ganz gewöhnlichen Tippfehlern. Da dies regelmäßig passiert, verfügen Rechnersysteme über entsprechende Korrekturmöglichkeiten. Das Edit-System Ihres TI-99/4A ermöglicht Ihnen das Schreiben und die Korrektur von Programmen, die sich im Arbeitsspeicher befinden. Jede Programmanweisung muß mit einer Nummer versehen werden. Die Anweisungen werden in aufsteigender Folge zu einem vollständigen Programm zusammengefaßt. Damit haben Sie die Möglichkeit, Zeilen einzufügen, vorausgesetzt, es ist noch Platz dafür vorhanden. Wegen der zu erwartenden Korrekturen empfiehlt es sich, die Numerierung der Anweisungen gleich mit einer größeren Schrittweite vorzunehmen, zum Beispiel gleich in Zehnerschritten zu numerieren. Dies können Sie übrigens auch dem Kommando NUMBER überlassen, falls Sie keine andere Schrittweite wünschen. Das NUMBER-Kommando erzeugt automatisch die fortlaufende Numerierung, so daß damit das Schreiben von Programmtexten erheblich vereinfacht wird. Wenn Sie zwischen zwei Zeilen mehr Korrekturen einfügen müssen als Platz verfügbar ist, so muß der ganze Programmtext neu durchnumeriert werden. Dies besorgt das Kommando RESEQUENCE, das dabei auch die Nummern in den Sprunganweisungen wie GOTO, GOSUB oder nach THEN verändert.

Korrekturen innerhalb einer Zeile sind mit den Funktionstasten für Delete (FCTN 1 und FCTN 2) sowie durch Überschreiben möglich. Ganze Zeilen können gelöscht werden, indem man nur die Zeilennummer tippt oder indem man im Korrekturmodus den Zeileninhalt mittels Erase (FCTN 3) löscht und danach entweder ENTER drückt oder zur nächsten Zeile weitergeht. Diesen Korrekturmodus erreicht man sowohl im TI-BASIC als auch im Extended BASIC durch Eingabe einer Zeilennummer und anschließendes Drücken der Taste mit dem Pfeil nach unten (FCTN X). Mit Pfeil nach unten oder Pfeil nach oben wird jeweils die folgende oder die vorhergehende Zeile angezeigt und zur Korrektur angeboten. Damit ist es auch möglich, verhältnismäßig schnell größere Bereiche eines Programmes zu löschen. Mit Pfeil nach unten holt man sich die erste zu löschende Zeile, und mit fortgesetztem Erase (FCTN 3) und nächste Zeile Holen (FCTN X) wird der

gewünschte Bereich gelöscht. In Extended BASIC ist zusätzlich noch folgendes Verfahren möglich: Zum Duplizieren einer Anweisung wird nach dem ENTER die Redo-Taste (FCTN 8) gedrückt. Es kann jetzt auch die Zeilennummer verändert werden. Durch Änderung der Nummer erhält man so ein Duplikat der letzten Anweisung. War man zuvor im NUMBER-Modus, so ergibt das Redo eine leere Zeile mit der zuletzt angezeigten Nummer. Mit Eintippen der letzten bzw. der Nummer der zu duplizierenden Zeile und Pfeil nach unten wird die gewünschte Zeile angezeigt. Wenn jetzt ENTER und gleich anschließend Redo gedrückt wird, steht die gesamte Zeile, also auch die Zeilennummer, zur Änderung zur Verfügung. Nach dieser Methode können auch ganze Zeilen verlagert werden. Es wird zunächst die Nummer der zu verlagernden Zeile getippt, dann Pfeil nach unten, ENTER, Redo, Zeilennummer ändern und ENTER. Damit ist die ursprüngliche Zeile dupliziert. Durch Eingabe der Nummer der Originalzeile und nachfolgendem ENTER wird diese gelöscht, und der Verlagerungsvorgang ist abgeschlossen.

Benutzung des Kassettenrecorders

Wie schon zuvor erwähnt, ist die Anschaffung des Originalkabels für den Anschluß eines Kassettenrecorders sehr zu empfehlen. Am besten ist ein Recorder geeignet, der über Fernbedienung und ein Bandzählwerk verfügt. Falls Ihr Recorder kein Zählwerk hat, können Sie sich recht gut mit folgendem Verfahren behelfen: Die Aufzeichnung eines Programms beginnt stets mit einigen Sekunden Vorspulen, danach folgt ein Dauerton von konstanter Frequenz und dann die eigentliche Information in akustischer Verschlüsselung. Die meisten Recorder verfügen über ein eingebautes Mikrofon. Sie können dies dazu nutzen, vor jedes auf Band abgespeicherte Programm ein paar Informationen über das folgende Programm zu sprechen. Wenn Sie danach kurz auf das Mikrofon klopfen, haben Sie eine akustische Markierung der Stelle, an der Sie beim Suchen das Gerät anhalten müssen.

Fernbedienung

Falls Sie an Ihrem TI-99/4A eine Fernbedienung haben, achten Sie stets darauf, daß die Taste „ALPHA LOCK“ bei Benutzung entriegelt ist. Die Fernbedienung funktioniert sonst nicht in allen Richtungen.

Benutzung von Steuerzeichen

Neben den Zeichen, die auf dem Bildschirm sichtbar dargestellt werden, gibt es noch eine Reihe weiterer Zeichen, die in der EDV allgemein verwendet werden und eine festgelegte Bedeutung haben. Diese Zeichen heißen Steuerzeichen oder Control Codes und dienen, wie der Name sagt, der Funktionssteuerung bei der Datenübertragung. Dies betrifft sowohl die Übertragung von Daten zwischen Rechnern und direkt angeschlossenen Geräten als auch die Datenfernübertragung über Telefonleitungen etc. Eine vollständige Tabelle der ASCII-Codes einschließlich der Steuerzeichen finden Sie im Anhang. Mit der Funktion CHR$ können Sie die Zeichen erzeugen und ggf. an externe Geräte senden. Darüber hinaus können die Control Codes auch für die Steuerung des Programmablaufes geeignete Hilfsmittel sein. Mit dem Unterprogramm KEY können Sie alle Codes feststellen, die über die Tastatur eingegeben werden. Jeder Taste der Tastatur sind mehrere Bedeutungen zugeordnet. In der Normalstellung gelten die unteren Gravierungen, und bei den Buchstaben gelten die kleinen. Wird die SHIFT-Taste gedrückt, dann gelten die oberen Gravierungen bzw. die Großbuchstaben. Darüber hinaus ist bei Standard-EDV-Terminals noch eine dritte Bedeutung vorhanden, die man analog zur SHIFT-Taste durch gleichzeitiges Drücken der CTRL-Taste und einer weiteren Taste erhält. Speziell beim TI-99/4A gibt es noch einige weitere interne Codes, die mittels der FCTN-Taste erzeugt werden.

Kapitel 2 Spielen

Um mit einem so leistungsfähigen Gerät, wie es auch ein kleiner Heim-Computer ist, entsprechend umgehen zu können, muß man ihn beherrschen lernen. Es gibt viele Wege, dieses Ziel zu erreichen: Man kann die technischen Unterlagen, Handbücher und das, was sonst alles an Beschreibungen mit dem Rechner geliefert wurde, systematisch durcharbeiten, also mehr oder weniger auswendig lernen. Es ist ein Verfahren, das von einiger Gründlichkeit zeugt, aber doch sehr oft mit nachlassender Begeisterung einhergeht. Warum eigentlich soll der Umgang mit Computern – bei aller Ernsthaftigkeit – nicht auch Spaß machen? Und gerade für Sie, lieber Leser, der Sie sich vielleicht zum erstenmal mit einem Computer auseinandersetzen, ist es wichtig, daß Sie beides erreichen: Ihren Computer zu beherrschen und daran auch einigen Spaß zu haben.

Nach den Vorbereitungen im vorigen Kapitel ist in diesem unter dem Motto Spielen stehenden Kapitel das Ausprobieren und spielerische Kennenlernen des TI-99/4A das Ziel. Es geht dabei nicht vorrangig um die Programmierung von Spielen, sondern um die „Lernmethode Spielen“, nämlich neue, wenig oder bisher noch nicht bekannte Dinge zunächst einmal mehr oder weniger zweckfrei auszuprobieren, also ganz einfach zu schauen: „Was passiert denn, wenn ich ...?“ Absicht dieses Kapitels, eigentlich des ganzen Buches, ist es, Sie zu ermuntern und anzuregen, Ihren TI-99/4A spielerisch locker zu erobern. Sie werden schon in diesem Kapitel an den etwas ausführlicheren Beispielen sehen, daß Spielen in diesem Sinne auch seinen Einsatz verlangt und Spielen nicht gleich verspielt ist.

Übrigens, es ist auch unter EDV-Profis eine gängige Redensart, wenn es darum geht, ein neues Gerät oder Programm kennenzulernen und sich darüber ein Urteil zu bilden: „Damit muß ich erst mal spielen.“ Wie es die Autoren dieses Buches mit dem TI-99/4A auch gemacht haben!

Einige der Beispiele in diesem Kapitel wurden ausgewählt, um ihnen

Lösungen für Situationen aufzuzeigen, die beim Programmieren immer wieder auftauchen. Sie sollen Ihnen als Programmierrezepte dienen oder, um es im Coputer-Jargon zu sagen, sie sollen Tools in Ihrer Programmier-Toolbox sein, Werkzeuge für den Programmierer in seiner Werkzeugkiste.

GRUNDLAGEN (TOOLBOX)

Anhand konkreter Aufgabenstellungen wollen wir einige Lösungsvarianten erarbeiten und dabei die benutzten Anweisungen des TI-BASIC und des Extended BASIC erläutern.

Beim Drucken von Ergebnissen oder bei der Verfolgung des Programmablaufes wird oft eine Pause gewünscht, um die Ausgabe genauer anschauen zu können. Die Länge der Pause soll entweder fest im Programm vorgegeben werden oder von außen durch Drücken einer Taste bestimmt werden. Ein ähnliches Problem hat man beim Ausgeben langer Ergebnislisten auf dem Bildschirm. Bei der PRINT-Anweisung schreibt der Rechner in die unterste Bildschirmzeile (Zeile 24), schiebt aber zuvor das ganze Bild um eine Zeile höher. Wenn also die Ausgabe länger als 20 Zeilen (wegen der READY- bzw. DONE-Meldung) ist, verschwindet der Anfang am oberen Bildschirmrand, bevor der Rest überhaupt ausgegeben wurde. In diesem Fall braucht man also ein Programm, das dafür sorgt, daß auf Wunsch die Ausgabe angehalten wird. Die bereits ausgegebenen Zeilen können dann bis zur Eingabe einer Quittung in Ruhe studiert werden.

In den beiden BASIC-Varianten des TI-99/4A gibt es Unterprogramme, die es dem Programmierer gestatten, eigene Zeichen zu gestalten. Wir werden damit ein paar Versuche durchführen und im Abschnitt über die Farben wieder darauf zurückkommen.

Die Programmierung von Pausen und Verzögerungen

1. Verzögerung um eine bestimmte Zeit (Pause)

Dummy-Schleife

In die Verarbeitungsschleife wird eine FOR-Anweisung eingefügt, die den Computer eine Weile beschäftigt. Je nach gewünschter Länge der Pause kann das eine sogenannte „leere FOR-Schleife“ sein oder eine

FOR-Schleife mit einer rechenintensiven Aufgabe. Als solche rechenintensiven Aufgaben eignen sich die mathematischen Funktionen recht gut, wie zum Beispiel SIN, COS, ATN, EXP usw. Die mathematische Bedeutung dieser Funktionen ist in diesem Zusammenhang ohne Belang, es kommt hier nur darauf an, daß ihre Berechnung einen gewissen Zeitaufwand erfordert. Die Argumente einiger Funktionen und ihre Ergebnisse können unter Umständen zu Fehlern im Programmablauf führen. Falls Ihnen der mathematische Hintergrund momentan nicht geläufig ist, benutzen Sie am besten die unten aufgeführten Formen, oder probieren Sie Ihre Variante erst einmal „trocken“ aus.

Die Länge der Pause wird durch den Endeparameter bestimmt. Probieren Sie zunächst einmal die leere FOR-Schleife mit verschiedenen Endeparametern aus, und stoppen Sie die Zeiten, um einen Anhalt für die erreichten Verzögerungszeiten zu bekommen:

```
100 PRINT "BEGINN DER PAUSE"
110 FOR DELAY = 1 TO 1000
120 NEXT DELAY
130 PRINT "ENDE DER PAUSE"
```

Im Extended BASIC kann die Pausenschleife auch in einer Zeile zusammengefaßt werden:

```
100 PRINT "BEGINN DER PAUSE"
110 FOR DELAY = 1 TO 1000 : : NEXT DELAY
120 PRINT "ENDE DER PAUSE"
```

Mit dem TO-Parameter gleich 1000 erhält man eine Verzögerungszeit von ca. 5 Sekunden. Durch Abänderung dieses Parameters kann die Zeit entsprechend verkürzt oder verlängert werden.

Wir fügen jetzt in der FOR-Schleife noch eine rechenintensive Anweisung ein, zum Beispiel X = SIN (1.23):

```
100 PRINT "BEGINN DER PAUSE"
110 FOR DELAY = 1 TO 100
115 X = SIN (1.23)
120 NEXT DELAY
130 PRINT "ENDE DER PAUSE"
```

oder im Extended BASIC auch:

```
100 PRINT "BEGINN DER PAUSE"
105 FOR DELAY = 1 TO 100 : : X = SIN (1.23) : : NEXT DELAY
130 PRINT "ENDE DER PAUSE"
```

Obwohl der Endeparameter nur noch 100 ist, dauert das Abarbeiten der Schleife länger als bei der leeren FOR-Schleife, nämlich rund 9 Sekunden.

Die leere FOR-Schleife braucht für 1000 Durchläufe 5 sec, also für einen Durchlauf $^5/_{1000}$ sec = 5 msec (1 msec = 1 Millisekunde = $^1/_{1000}$ Sekunde). Die FOR-Schleife mit dem SIN braucht für 100 Durchläufe 9 sec, also für einen Durchlauf $^9/_{100}$ sec = $^{90}/_{1000}$ sec = 90 msec. Zieht man davon die Dauer für die leere Schleife ab, so erhält man die Zeit für die Berechnung der SIN-Funktion mit 85 msec.

Damit haben wir auch gleich ein Verfahren zum Messen von Rechenzeiten. Besonders bei einzelnen Anweisungen oder kleinen Programmstücken sind die tatsächlichen Rechenzeiten so kurz, daß sie mit einer Stoppuhr nicht gemessen werden können. Man baut dann die interessierende Anweisung oder das betreffende Programmstück in eine FOR-Schleife ein und läßt es so oft wiederholen, bis eine gut meßbare Zeit herauskommt. Die Division durch die Anzahl der Wiederholungen ergibt die Ausführungszeit für die auszumessende Probe. Von der so bestimmten Zeit muß aber noch die Zeit für die Ausführung der leeren FOR-Schleife abgezogen werden, die man auf die gleiche Art ermittelt.

Benutzung des Unterprogrammes SOUND

Mit CALL SOUND (D, F, L) können Töne und Geräusche erzeugt werden, die durch Dauer, Frequenz und Lautstärke bestimmt werden. Für den hier interessierenden Zweck benötigen wir nur den Parameter für die Dauer und setzen deshalb die beiden anderen Parameter auf solche Werte, daß sie nicht stören, also z.B. F = 30000 und L = 30. Für die Dauer D können Werte von 1 bis 4250 angegeben werden, die ungefähr Millisekunden entsprechen, also 0.001 bis 4.250 sec. Während der Computer, durch das Unterprogramm SOUND veranlaßt, den gewünschten Ton mit der durch D festgelegten Länge abspielt, läuft das Programm weiter. Lediglich wenn ein weiterer Ton ausgegeben werden

soll, hält das Programm so lange an, bis der erste Ton beendet ist. Dies machen wir uns in den beiden folgenden Anweisungen zunutze:

```
200 CALL SOUND (2000, 30000, 30)
210 CALL SOUND (1, 30000, 30)
```

Mit dem ersten Aufruf wird ein Ton von ca. 2 sec Dauer erzeugt. Wegen der Frequenz von 30000 Hz und der möglichst geringen Lautstärke ist natürlich nichts zu hören. Der zweite Aufruf erzeugt nochmals den gleichen Ton, allerdings nur von ca. 1 msec Dauer, jedoch sorgt dieser zweite Aufruf dafür, daß die weitere Programmausführung so lange angehalten wird, bis der erste Ton vollständig abgespielt wurde, also eine Pause von rund 2000 msec = 2 sec eingehalten wurde.

Durch diese beiden Anweisungen können Verzögerungen von einigen Millisekunden bis etwas über 4 Sekunden erzeugt werden. Für längere Verzögerungen kann folgendes Delay-Programm benutzt werden:

```
110 FOR DELAY = 1 TO ANZSEC
120 CALL SOUND (1000, 30000, 30)
130 CALL SOUND (1, 30000, 30)
140 NEXT DELAY
```

Mit dem Parameter ANZSEC wird die Verzögerungsdauer in Sekunden angegeben. Eventuell muß im ersten Aufruf von SOUND der Wert 1000 etwas verändert werden, damit sich für jeden Durchlauf genau eine Sekunde ergibt. Geringe Unterschiede in der Dauer sind möglich und müssen durch Ausprobieren ermittelt werden.

2. Verzögerung um eine von außen vorgegebene Zeit

Mittels INPUT und ACCEPT

An der Stelle, an der das Programm auf unbestimmte Zeit angehalten werden soll, wird eine Eingabe-Anweisung eingefügt. Normalerweise dienen diese Anweisungen dazu, während des Programmlaufs Variablen mit Werten zu versorgen. Die Anweisungen INPUT und ACCEPT halten die Programmausführung so lange an, bis der angeforderte Wert über die Tastatur eingetippt wurde und durch Drücken der ENTER-Taste zur weiteren Bearbeitung an den Computer übergeben wurde.

Entscheidend ist hier also, daß die Programmausführung bis zum Drükken der ENTER-Taste angehalten wird. Als Variable, die mit Werten versorgt werden soll, setzen wir eine sonst weiter nicht benötigte Variable ein, eine sogenannte dummy Variable. Wenn man dafür eine Zeichenvariable nimmt, kann es zu keinen Eingabefehlern führen, da praktisch alle Zeichen als Eingabe erlaubt sind. Auch die Eingabe eines Leerzeichens (Leertaste) oder die Quittierung der Input-Anweisung mit Return (Drücken der ENTER-Taste) ist erlaubt.

Wir können unser Pausenprogramm jetzt wie folgt schreiben:

```
100 PRINT "BEGINN DER PAUSE"
110 INPUT DUMMY$
120 PRINT "ENDE DER PAUSE"
```

Der Nachteil dieses Verfahrens: Es erscheint ein Fragezeichen als Eingabe-Aufforderung auf dem Bildschirm, und das gesamte Bild wird nach oben weitergeschoben.

Diese Verschiebung kann im Extended BASIC durch die ACCEPT-Anweisung vermieden werden:

```
110 ACCEPT AT (1,1) : DUMMY$
```

Mit der AT-Funktion kann der Cursor an jeder beliebigen Stelle des Bildschirms positioniert werden. An der durch Zeilennummer und Spaltennummer bestimmten Stelle blinkt dann der Cursor so lange, bis eine mit ENTER abgeschlossene Eingabe erfolgt ist. Wenn eine leere Eingabe erfolgt, also nur die ENTER-Taste gedrückt wird, ist der Bildschirm danach unverändert.

Mittels Unterprogramm KEY

Die zuvor erwähnten Nachteile (Verschieben des Bildes durch die Input-Anweisung und erforderliche Quittierung einer Eingabe mit ENTER) werden durch die Benutzung von

```
CALL KEY (0, Z, S)
```

vermieden. Das Unterprogramm KEY gibt es sowohl im TI-BASIC als auch im Extended BASIC. Der erste Parameter 0 legt fest, daß eine

Eingabe von der Konsolentastatur erwartet wird. Das KEY-Unterprogramm „schaut" nur auf die Tastatur. Es meldet jede gedrückte Taste, ohne daß die ENTER-Taste betätigt werden muß. Es erfolgt keine Eingabe-Aufforderung wie bei INPUT mit dem Fragezeichen oder bei ACCEPT mit dem blinkenden Cursor und auch keine Ausgabe des gelesenen Zeichens auf dem Bildschirm. In Z wird der dem eingegebenen Zeichen entsprechende numerische Wert des ASCII-Codes und in S der Status abgelegt. Für unseren Zweck brauchen wir nur die Statusmeldung. Solange S = 0 ist, heißt das, daß keine Taste gedrückt wurde. Damit ergibt sich für unser Verzögerungsprogramm:

```
100 PRINT "BEGINN DER PAUSE"
110 CALL KEY (0, Z, S)
120 IF S = 0 THEN 110
130 PRINT "ENDE DER PAUSE"
```

Wir erhalten damit eine Endlos-Schleife, die durch das Drücken einer beliebigen Taste abgebrochen wird. Hierdurch erhält die Statusvariable den Wert S = 1 oder S = −1, also ungleich 0, und die Verarbeitung wird in der nächsten Zeile fortgeführt.

Die Steuerung von langen Ausgaben

Ähnlich wie beim LIST-Kommando wollen wir erreichen, daß die Ausgabe einer langen Liste durch Drücken einer Taste angehalten werden kann, und daß mit der Ausgabe nach einem weiteren Tastendruck fortgefahren wird. Dazu benutzen wir wieder das KEY-Unterprogramm, das allein in der Lage ist festzustellen, ob eine Taste gedrückt wurde. Wir simulieren das Programm, das die lange Liste ausgibt, durch eine einfache FOR-Schleife:

```
100 FOR I = 1 TO 100
..
..
..
200 PRINT I
..
..
..
400 NEXT I
```

Wenn Sie dieses Programm starten, werden hintereinander die Zahlen von 1 bis 100 in jeweils eine neue Zeile geschrieben, und Sie haben keine Möglichkeit, die Ausgabe für eine Weile anzuhalten und wieder fortzusetzen (außer etwas massiv durch ein Break, das durch Drücken der Funktionstaste CLEAR erzeugt wird). Mit dem Kommando CONTINUE wird der Weiterlauf des Programms veranlaßt. Dieses Verfahren hinterläßt auf dem Bildschirm eine durch die Meldungen

```
* BREAKPOINT IN xxx
> CONTINUE
```

unterbrochene Liste. Wir zeigen Ihnen zwei Verfahren, die Listenausgabe ohne Veränderungen des Listenbildes „zu bremsen". Beim ersten wird die Ausgabe durch Drücken der Taste H (wie Halt) angehalten und durch Drücken der Taste W (wie weitermachen) fortgesetzt. Bei der zweiten Variante kann die Ausgabe durch Drücken einer beliebigen Taste angehalten und ebenso durch Drücken einer beliebigen Taste wieder aufgenommen werden.

In beiden Varianten werden zwei KEY-Aufrufe benötigt. Der erste dient dazu festzustellen, ob eine Taste gedrückt wurde. Wenn nicht, dann muß das „Bremsprogramm" übersprungen werden, wenn ja, dann wird eine Schleife gestartet, aus der nur nach Erfüllen einer geeigneten Bedingung wieder herausgesprungen werden kann.

1. Variante:

```
100 FOR I = 1 TO 100
200 PRINT I
300 CALL KEY (0, Z, S)
310 IF S = 0 THEN 360
320 IF CHR$ (Z) <> "H" THEN 360
330 CALL KEY (0, Z, S)
340 IF S = 0 THEN 330
350 IF CHR$ (Z) <> "W" THEN 330
360 REM
400 NEXT I
```

Die Zeilen 300 bis 360 stellen das Bremsprogramm dar, wobei die Zeile 360 REM als Sprungziel dient. Der erste KEY-Aufruf stellt zunächst einmal fest, ob überhaupt eine Taste gedrückt wurde. Wenn das nicht

der Fall ist, dann ist der Status S = 0, und die IF-Abfrage in der Zeile 310 sorgt dafür, daß das Bremsprogramm übersprungen wird. In der nächsten Zeile, die ja nur ausgeführt wird, wenn eine Taste gedrückt wurde, wird kontrolliert, ob dies die H-Taste war. Wenn nein, dann wird ebenfalls nach 360 gesprungen. Nur wenn die H-Taste gedrückt wurde, wird die Zeile 330 erreicht. Zusammen mit Zeile 340 stellt sie eine Endlosschleife dar, aus der man nur durch die Bedingung S <> 0 herauskommt. Mit S <> 0 gelangt man in die Zeile 350, in der kontrolliert wird, ob der Sprung aus der Schleife der Zeilen 330 und 340 durch die W-Taste verursacht wurde. Wenn nein, dann wird wieder in die Schleife zurückgesprungen, wenn ja, dann kann das unterbrochene Verarbeitungsprogramm ab Zeile 360 fortgeführt werden.

Die Abfragen in den Zeilen 320 und 350 benutzen die Funktion CHR$. Hierdurch wird das ASCII-Zeichen, das dem in Z enthaltenen numerischen Wert entspricht, ermittelt. Wird auf eine solche Umwandlung verzichtet, so könnten die beiden Zeilen auch folgendermaßen geschrieben werden:

```
320 IF Z <> 72 THEN 360
```

und

```
350 IF Z <> 87 THEN 330
```

In der ASCII-Code-Tabelle oder durch die Anweisung

```
PRINT ASC ("H") ; ASC ("W")
```

erkennen Sie, daß dem Zeichen H der Wert 72 und dem W der Wert 87 zugeordnet ist. Wir glauben, daß die erste Lösung mit CHR$ die bessere ist, da diesen Anweisungen sofort anzusehen ist, was geschehen soll. Es ist deutlich zu erkennen: „Ich will wissen, ob die gedrückte Taste die H-Taste war!“ Es ist also eine Frage des Programmierstils – und gute Programme zeichnen sich dadurch aus, daß man aus ihnen möglichst klar und einfach erkennt, was sie machen.

2. Variante:

```
100 FOR I = 1 TO 100
200 PRINT I
300 CALL KEY (0, Z, S)
```

```
310 IF S = 0 THEN 360
340 CALL KEY (0, Z, S)
350 IF S = 0 THEN 340
360 REM
400 NEXT I
```

Die Zeiten 300 und 310 funktionieren genauso wie in der ersten Variante. Nur wird diesmal nicht abgefragt, welche Taste gedrückt wurde, sondern die Verarbeitung wird mit den Zeilen 340 und 350 fortgeführt, die die aus der ersten Variante bekannte Aufgabe haben. Auch diese Schleife wird ohne Überprüfung des gelesenen Zeichens verlassen. Im Unterschied zur ersten Variante kann hier eine beliebige Taste gedrückt werden, um die Ausgabe anzuhalten und um sie wieder weiterlaufen zu lassen. Da aber zwischen den beiden KEY-Aufrufen nur eine kurze Zeitspanne zur Verfügung steht, muß die gedrückte Taste wieder sehr schnell losgelassen werden. Andernfalls wird die FOR-Schleife sofort weiter ausgeführt, und der Effekt ist gleich Null. Um das zu vermeiden, schließen wir eine kleine Verzögerung ein:

```
100 FOR I = 1 TO 100
200 PRINT I
300 CALL KEY (0, Z, S)
310 IF S = 0 THEN 360
320 CALL SOUND (200, 33333, 30)
330 CALL SOUND (1, 33333, 30)
340 CALL KEY (0, Z, S)
350 IF S = 0 THEN 340
360 REM
400 NEXT I
```

Diese Verzögerung von etwa $^2/_{10}$ Sekunden reicht im allgemeinen aus, um die Ausgabe sicher anzuhalten. Sie hat auch noch einen anderen, schönen Nebeneffekt: Sie bewirkt ein Art "slow motion" der Listenausgabe, wenn eine Taste ständig gedrückt wird.

Die Gestaltung eigener Zeichen

Die Festlegung der Form von Zeichen erfolgt beim TI-99/4A in einem Raster von 8 mal 8 Punkten. Hierbei wird für jeden Punkt festgelegt,

welche Farbe er haben soll. Diese beiden Farbarten sind die Hinter- und die Vordergrundfarbe. Alle Punkte, für die keine andere Festlegung getroffen wurde, haben die Hintergrundfarbe. Man braucht jetzt noch ein Verfahren, das sagt, welche Punkte in der Vordergrundfarbe dargestellt werden sollen. Das ganze Raster wird dazu in acht Zeilen eingeteilt, von denen jede eine linke und eine rechte Hälfte mit jeweils vier Punkten hat. Für diese vier Punkte setzt man die Werte 0 oder 1 ein. Bei 0 gilt die Hintergrundfarbe und bei 1 die Vordergrundfarbe. Man erhält somit eine Codierung, die aus Folgen von 0 und 1 besteht. Da alle 64 Punkte so beschrieben werden müssen, würde die Folge ziemlich lang. Es gibt eine kürzere Schreibweise dafür, nämlich die Hexadezimalziffern, deren Bedeutung im nächsten Kapitel ausführlicher behandelt wird.

Das Codierungsschema sieht folgendermaßen aus:

Binär	Hexadezimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Abb. 2.1: Codierungsschema für die Gestaltung eigener Zeichen

Nach diesem Schema sind auch die standardmäßig vorhandenen Zeichen codiert. In Extended BASIC gibt es das Unterprogramm CHARPAT, mit dem die Muster-Codierung sowohl der standardmäßig vor-

handenen als auch der selbst definierten Zeichen abgefragt werden kann. Im Anhang finden Sie diese Patternliste für den Standardzeichensatz abgedruckt. Mit dem folgenden Programm können Sie die Liste selbst erzeugen (dies ist allerdings nur im Extended BASIC möglich):

```
100 REM CHARACTER PATTERN AUSDRUCKEN
110 REM = = = = = = = = = = = = = = = = = = =
120 REM
130 FOR I = 32 TO 143
140 CALL CHARPAT (I, CH$)
150 PRINT I;"I "; CHR$ (I);" I "; CH$
160 NEXT I
```

In diesem Programm können wir die zuvor besprochene „Listenbremse“ anwenden:

```
100 REM CHARACTER PATTERN AUSDRUCKEN
110 REM = = = = = = = = = = = = = = = = = = =
120 REM
130 FOR I = 32 TO 143
140 CALL CHARPAT (I, CH$)
150 PRINT I;"I "; CHR$(I);" I "; CH$
160 CALL KEY (0, Z, S)
170 IF S = 0 THEN 220
180 IF CHR$ (Z) <> "H" THEN 220
190 CALL KEY (0, Z, S)
200 IF S = 0 THEN 190
210 IF CHR$ (Z) <> "W" THEN 190
220 NEXT I
```

In dieser Liste erscheinen spaltenweise die Nummer des Zeichens, das Zeichen selbst und danach die Mustercodierung. Bei I = 100 hat die Liste einen Schönheitsfehler, weil die Werte dreistellig werden und das Listenbild um eine Stelle weiter rückt. Eine “Schönheitsreparatur“ erfordert, daß die Zahlen unabhängig von ihrer Größe immer dreistellig geschrieben werden. Für diesen Zweck gibt es die Möglichkeit der Formatierung des ausgedruckten Textes mittels der PRINT USING-

Anweisung. In einer weiteren Anweisung, der IMAGE-Anweisung, beschreiben wir das Ausgabeformat, wobei das Zeichen # für jeweils ein Ausgabezeichen steht. Das modifizierte Programm sieht dann wie folgt aus:

```
100 REM CHARACTER PATTERN AUSDRUCKEN
110 REM = = = = = = = = = = = = = = = = = = =
120 REM
130 FOR I = 32 TO 143
140 CALL CHARPAT (I, CH$)
150 PRINT USING 155: I, CHR$ (I), CH$

155 IMAGE ### I # I ################
160 CALL KEY (0, Z, S)
170 IF S = 0 THEN 220
180 IF CHR$ (Z) <> "H" THEN 220
190 CALL KEY (0, Z, S)
200 IF S = 0 THEN 190
210 IF CHR$ (Z) <> "W" THEN 190
220 NEXT I
```

In dieser Liste werden Sie feststellen, daß ab I = 127 die Patterns nur noch 0 enthalten. Das Zeichen 127 nimmt im ASCII-Code eine besondere Funktion ein. Es ist das Delete-Zeichen und wird üblicherweise, wie der Name sagt, zum Entfernen von Zeichen benutzt, es ist sozusagen der Radiergummi. Auf Fernschreibern und Druckern wird es meist als ein vollausgefülltes Zeichen realisiert, mit dem man ein anderes vollständig überdrucken kann. Diese besondere Funktion des Delete-Zeichens wird auf dem Bildschirm nicht benötigt, da sich jedes Zeichen leicht durch ein anderes überschreiben läßt.

Wenn wir eigene neue Zeichen definieren, ist es zweckmäßig, sie auf die Positionen 127 bis 143 zu legen. Beim Ausdrucken bleiben sie dann auch nach Beendigung des Programmlaufs erhalten. Benutzen wir dagegen Positionen aus dem Bereich von 32 bis 126, so ändern sich die selbstdefinierten Zeichen nach Beendigung des RUN wieder in ihre standardmäßig festgelegte Form. Probieren wir es einmal aus:

In dem eingebauten Pattern haben die kleinen Buchstaben ein etwas kleineres Format als die Großbuchstaben, haben aber die gleiche Form.

Wir wollen uns deshalb ein eigenes kleines a definieren:

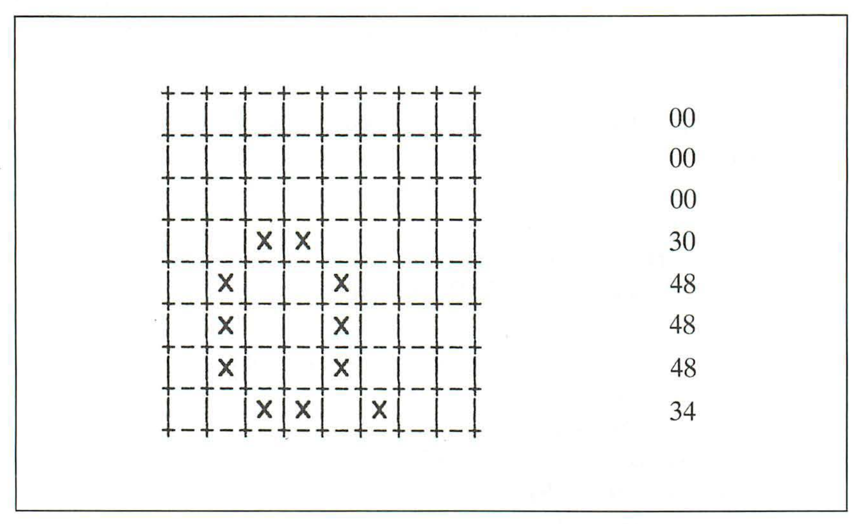


Abb. 2.2: Codierungsschema für den Buchstaben Klein-a

Der Patterncode lautet also „0000003048484834". Dieses Pattern bringen wir jetzt an die Stelle des kleinen a. Aus der ASCII-Tabelle entnehmen wir, daß dies die Position 97 ist. Das Unterprogramm CHAR ermöglicht es, solche Patterns einzubringen. Wir schreiben uns dazu das folgende kleine Programm:

```
100 A$ = "0000003048484834"
110 CALL CHAR (97, A$)
120 PRINT "Aa"
```

Das selbstdefinierte kleine a blitzt nur kurz auf, und wir sehen wieder das bereits bekannte verkleinerte Groß-A. Die eigenen Muster werden auf den standardmäßig belegten Positionen nur während der Programmausführung angezeigt. Wir müssen deshalb die Laufzeit entsprechend verlängern. Mit der Anweisung

```
130 GOTO 130
```

erzeugen wir eine Endlosschleife, die nur durch CLEAR abgebrochen werden kann. Solange dieses Programm läuft, werden alle a auf dem

Bildschirm in dieser Form dargestellt. Nach dem obigen Muster dürfte es Ihnen nicht mehr schwerfallen, weitere Buchstaben zu generieren.

Dieses Verfahren wollen wir jetzt verwenden, um ein Blinkzeichen zu erzeugen, dessen Form dabei invertiert wird, also die Stellen, die hell sind, werden dunkel und umgekehrt.

```
100 A$ = "FFFFC3C3C3C3FFFF"
110 B$ = "00003C3C3C3C0000"
120 CALL CHAR (130, A$)
130 CALL CHAR (131, B$)
140 CALL CLEAR
150 CALL HCHAR (10, 10, 130, 1)
160 CALL SOUND (300, 33333, 30)
170 CALL SOUND (1, 33333, 30)
180 CALL HCHAR (10, 10, 131, 1)
190 CALL SOUND (300, 33333, 30)
200 CALL SOUND (1, 33333, 30)
210 GOTO 150
```

Mit den SOUND-Aufrufen wird in bekannter Weise die Blinkdauer bestimmt. Im Programm enthalten ist natürlich wieder eine Endlosschleife (Zeilen 150 bis 210), die nur durch CLEAR beendet werden kann.

Das Ganze kann auch noch farbig gestaltet werden, worauf wir in dem Abschnitt über die Farben des TI-99/4A zusammen mit weiteren Anwendungen der CHAR-Routine eingehen werden. Im Zusammenhang mit den Plotprogrammen und dort speziell bei der hochauflösenden Graphik werden wir weitere Anwendungen des CHAR-Unterprogramms für die Erzeugung von Zeichen kennenlernen.

SPIELE MIT DEM ZUFALL

In vielen Spielen, wie beim Mischen der Karten oder beim Würfeln, ist der Zufall ein wichtiges Element. Auch in Modellen, die Vorgänge in der Technik, Wirtschaft, Naturwissenschaft usw. beschreiben, spielen durch den Zufall gesteuerte Ereignisse eine Rolle. In einem Modell, das etwa den Straßenverkehr in einer Stadt beschreibt, kann man Ort, Zeit

und Richtung eines Fahrzeugs als zufallsbestimmte Größen ansehen. Da solche Modelle mit Zufallselementen eine große praktische Bedeutung haben (man nennt solche Verfahren auch Monte-Carlo-Rechnungen), gibt es auch eine große Anzahl von Verfahren und Methoden, diese mit EDV zu bearbeiten.

In allen diesen Verfahren spielt die Erzeugung von zufälligen Zahlen eine wesentliche Rolle. Für praktisch alle Computer – natürlich auch für den TI-99/4A – gibt es Programme, die solche zufallsverteilten Zahlen (oder englisch random numbers) erzeugen. Die Funktion RND erzeugt Zufallszahlen zwischen 0 und 1.

Im Computer werden diese Zahlen im allgemeinen mit einem Programm erzeugt. Es gibt mathematische Verfahren, die eine Folge von Zahlen liefern, die die Eigenschaften von zufallsverteilten Zahlen haben. Diese Erzeugung von Randomzahlen per Programm hat noch die Eigenschaft, daß bei Bedarf stets die gleiche Zahlenfolge erzeugt werden kann, was für den Vergleich von Verfahren und für die Programmentwicklung von großem Wert sein kann.

Wir drucken uns als erstes einmal den Anfang der Folge von Zufallszahlen aus:

```
100 PRINT RND
110 GOTO 100
```

Die Folge beginnt

```
. 8225408469
.163164479
.4352797351
.3233738121 usw.
```

Jedesmal, wenn Sie dieses Programm starten, erhalten Sie die gleiche Folge. Um aber „echte" Zufallszahlen zu bekommen, also solche, deren Folge bei jedem RUN anders ist, setzen Sie als erste Anweisung ein RANDOMIZE davor. Diese Anweisung alleine sorgt dafür, daß mit jedem Programmverlauf eine vorher unbestimmbare Folge produziert wird. Es ist auch der Aufruf RANDOMIZE mit einem numerischen Argument möglich. Immer wenn der numerische Ausdruck gleich ist, wird auch die gleiche Folge erzeugt.

Probieren Sie das Programm

```
100 RANDOMIZE xxx
110 PRINT RND
120 GOTO 110
```

mit verschiedenen Werten für xxx aus, und lassen Sie es ganz weg.

Wie Sie gesehen haben, liegen die Zahlen alle im Bereich zwischen 0 und 1, genau 0 < = RND < 1. Diese Normierung der Randomzahlen auf diesen Wertebereich stellt keine besondere Einschränkung dar, da durch Multiplikation der erzeugten Zufallszahlen mit einem geeigneten Faktor und gegebenenfalls durch Addition jeder benötigte Wertebereich belegt werden kann.

Als erstes Beispiel entwickeln wir ein Programm, das sechs verschiedene Zahlen aus dem Bereich von 1 bis 49 „würfelt". Die Autoren übernehmen bei Verwendung dieser Zahlen als Lottozahlen keine Garantie für irgendeinen Erfolg. (Sollten Sie trotzdem einen größeren Gewinn erzielen, wird es bestimmt kein Problem sein, uns ausfindig zu machen.)

```
210 WURF = INT (RND * 48.5) + 1
220 PRINT WURF
230 GOTO 210
```

Wir erhalten eine Endlosfolge von Zahlen, die mit 40, 8, 22, 16, 9, 28, ... beginnt.

In der Zeile 210 wird der Zahlenbereich zwischen 0 und 1 durch Multiplikation mit 48.5 auf den Bereich zwischen 0 und 48.5 ausgedehnt. Die Funktion INT macht daraus ganze Zahlen zwischen 0 und 48, und durch die Addition mit 1 erhalten wir den gewünschten Bereich der ganzen Zahlen zwischen 1 und 49. Das Programm wird jetzt noch so ergänzt, daß genau sechs Zahlen erzeugt werden. Da bei jedem Wurf prinzipiell jede Zahl zwischen 1 und 49 erzeugt werden kann, ist es möglich, daß unser Verfahren die gleiche Zahl mehrmals würfelt. Wir müssen deshalb nach jedem Wurf kontrollieren, ob die soeben erhaltene Zahl bereits vorhanden ist. Falls dies zutrifft, muß der Wurf so oft wiederholt werden, bis eine noch nicht vorhandene Zahl gewürfelt wird.

Wir legen uns deshalb einen Vektor LZ mit Platz für 6 Werte an, in den wir lauter verschiedene Werte würfeln:

```
110 DIM LZ (6)
120 DEF WURF = INT (RND * 48.5) + 1
130 LZ (1) = WURF
140 WZ = 1
150 NW = WURF
160 FOR I = 1 TO WZ
170 IF LZ (I) = NW THEN 150
180 NEXT I
190 WZ = WZ + 1
200 LZ (WZ) = NW
210 IF WZ < 6 THEN 150
220 PRINT "DIE AUSSPIELUNG ERGAB "; "DIE LOTTOZAHLEN:"
230 FOR I = 1 TO 6
240 PRINT LZ (I);
250 NEXT I
```

In der Zeile 120 haben wir mit DEF eine Funktion definiert, die jeweils eine Zahl zwischen 1 und 49 „würfelt“. Diese selbstdefinierte Funktion WURF kann in Ausdrücken benutzt werden, genauso wie die bereits im System enthaltenen Funktionen (wie zum Beispiel RND).

Dieses Programm liefert natürlich immer die gleichen Zahlen. Erst durch die zusätzliche Anweisung

```
RANDOMIZE
```

erhalten Sie jedesmal einen anderen Satz von Lottozahlen – und das so oft, bis Sie die sechs richtigen haben.

Bestimmung der Kreiszahl π mittels Zufallszahlen

Nachdem wir mit dem Lottoprogramm für den Lebensunterhalt gesorgt haben, können wir uns in aller Ruhe zweckfreien Dingen, also richtigen Spielen zuwenden. Wußten Sie, daß man die Zahl π, die in der Kreisberechnung benötigt wird, auch durch Würfeln bestimmen kann?

Wir gehen von folgender Figur aus:

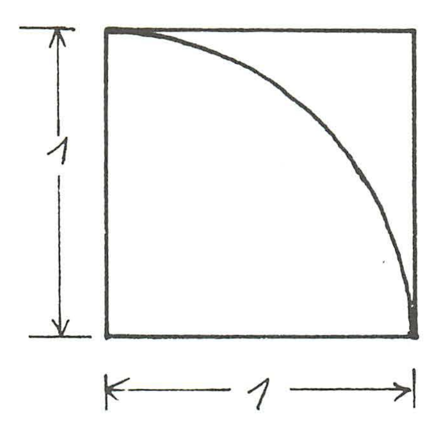


Abb. 2.3: Quadrat der Kantenlänge 1 mit Kreissegment

Wir haben ein Quadrat mit der Kantenlänge 1, in das ein Teil eines Kreises mit dem Radius 1 eingezeichnet ist. Die Figuren haben folgende Flächen:

Fläche des Quadrats: $F_q = 1 * 1 = 1$

Fläche des Kreissegmentes: $F_k = \pi * 1 * \frac{1}{4} = \pi/4$

Wir lassen jetzt auf diese Figur Regentropfen fallen, die zufallsverteilt sind. Wenn wir es lange genug regnen lassen, dann werden die Regentropfen auf der Figur gleichmäßig verteilt sein, das heißt, daß die Anzahl A_q der Tropfen, die auf das Quadrat fallen, der Fläche des Quadrats proportional ist und die Anzahl A_k der Tropfen, die in das Kreissegment fallen, der Fläche des Segments proportional ist. Es verhält sich also

$$\frac{A_k}{A_q} = \frac{F_k}{F_q}$$

Wenn wir für die Flächen die obigen Ergebnisse einsetzen, erhalten wir:

$$\frac{A_k}{A_q} = \frac{\pi}{4}$$

oder

$$\pi = \frac{4\ A_k}{A_q}$$

Wir brauchen also nur die Regentropfen zu zählen, um daraus π zu bestimmen. Wie zählen wir die Tropfen, das heißt, wie lassen wir unseren TI-99 regnen?

Unsere Figur hat die Kantenlänge 1. Die Zufallszahlen liegen in dem Intervall von 0 bis 1. Wenn wir also zwei Zufallszahlen hintereinander „würfeln" und diese x und y zuordnen, wobei x und y die Position im Quadrat angeben, auf die der Tropfen fällt, dann haben wir gerade eine „quadratische Regenwolke" bestimmt, die das Quadrat der Kantenlänge 1 gleichmäßig beregnet. Wenn wir dieses Programmstück in eine FOR-Schleife einbauen, können wir mit dem Endeparameter die Anzahl der Regentropfen festlegen, die auf das Quadrat fallen sollen. Als nächstes müssen wir noch mitzählen, wie viele von diesen Tropfen auch in das Kreissegment gefallen sind. Dies geschieht dadurch, daß wir den Abstand des Tropfens vom Nullpunkt, also von der linken unteren Ecke bestimmen. Dazu erinnern wir uns an den Satz des Pythagoras, mit dem wir finden, daß

$$r^2 = x^2 + y^2$$

ist oder

$$r = \sqrt{x^2 + y^2}$$

also r = SQRT (x * x + y * y)

Ist r kleiner als 1, dann ist der Tropfen in das Kreissegment gefallen. Wir können uns die Berechnung der Quadratwurzel ersparen, da die Wurzel aus Zahlen kleiner als 1 auch kleiner als 1 ist und uns hier nur die Frage: „Ist r kleiner als 1?" interessiert.

Das nachfolgende Programm führt das oben besprochene Verfahren aus und liefert auch gleich noch einige Werte zur Abschätzung der Genauigkeit:

```
100 PRINT "BESTIMMUNG DER ZAHL PI"
110 PRINT "MITTELS ZUFALLSZAHLEN"
```

```
120 PRINT
130 PRINT "ANZAHL DER WUERFE EINGEBEN ";
140 INPUT AQ
150 AK = 0
160 FOR I = 1 TO AQ
170 X = RND
180 Y = RND
190 R = X * X + Y * Y
200 IF R > = 1 THEN 220
210 AK = AK + 1
220 NEXT I
230 PIZUF = 4 * AK/AQ
240 DPI = PIZUF - 3.14159265
250 DPIREL = ABS (DPI)/PIZUF * 100
260 PRINT
270 PRINT
280 PRINT "PI PER ZUFALL = "; PIZUF
290 PRINT "DIE ABWEICHUNG VON "; DPI
300 PRINT "VOM TATSAECHLICHEN WERT ";
          "VON PI = 3.14159"
310 PRINT "BETRAEGT BEI "; AQ; "WUERFEN ";
          DPIREL; "‰"
320 END
```

Die zusätzlichen Angaben über die Abweichung vom tatsächlichen Wert gestatten rückwirkend eine Aussage über die Qualität der Zufallszahlen. Wenn diese Zahlen, wie behauptet, gleichverteilt sind und auch keine Korrelation mit den Nachbarzahlen vorhanden ist, also die durch die (x, y)-Paare bestimmten Orte über die Quadratfläche gleichmäßig verteilt sind, dann ist der durch Würfeln bestimmte Wert für π mit einem statistischen Fehler von $\pm\ 100/\sqrt{A_q}\ \%$ behaftet. Das heißt also, daß mit 100 Würfen π auf 10% genau bestimmt werden kann, und daß für eine Genauigkeit von 1% 10000 Würfe erforderlich sind.

Geläufigkeitsübung für die Tastatur des TI-99/4A

Um mit Ihrem TI-99/4A flüssig arbeiten zu können, ist es von wesentlicher Bedeutung, die Tastatur zu beherrschen, daß Sie also jede Taste auf Anhieb finden. Mit dem folgenden Programm schreiben wir uns

einen Trainer dafür: Das Programm bestimmt mittels Zufallszahl ein Zeichen, druckt es aus und wartet, bis Sie die betreffende Taste gedrückt haben. Die Zeit, die Sie dazu benötigen, wird gestoppt und die Anzahl der richtigen und falschen Eingaben mitgezählt.

Aus der ASCII-Tabelle sehen wir, daß den abdruckbaren Zeichen die Codewerte von 33 bis 126 zugeordnet sind. Zum Code 32 gehört das Leerzeichen (Zwischenraum, Blank).

Ähnlich wie beim Lottozahlen-Programm definieren wir uns mit DEF eine Funktion, die zufallsverteilte ganze Zahlen aus dem Bereich von 33 bis 126 liefert. Wenn nur die Großbuchstaben vorkommen sollen, müssen die kleinen Buchstaben (Codes 97 bis 122) in Großbuchstaben umgesetzt werden. Diese Umwandlung der kleinen in große Buchstaben (englisch upshift) besorgt die Anweisung in der Zeile 350. Wenn sowohl die kleinen als auch die großen Buchstaben getestet werden sollen, dann muß die Anweisung in Zeile 350 herausgenommen werden, entweder durch Löschen oder durch Vorsetzen von REM. Diese letzte Vorgehensweise hat den Vorteil, daß durch Entfernen des Wortes REM wieder der ursprüngliche Zustand hergestellt werden kann.

Das Programm schreibt zunächst eine Anfangsmeldung und fragt nach der Anzahl der Versuche. In der FOR-Schleife mit dem Parameter LOOP (Zeilen 330 bis 580) werden dann die Tests ausgeführt. Die Zeit bis zum Drücken einer Taste wird in der FOR-Schleife mit dem Parameter T gemessen (ab Zeile 440). Durch Versuche kann man herausbekommen, daß T/3 recht gut die Zeit in Zehntelsekunden ergibt. In Zeile 530 erhält man zunächst aus INT(T/3) die Zeit in Zehntelsekunden ganzzahlig und durch Division durch 10 dann den Wert in Sekunden mit einer Nachkommastelle. Die einzelnen Zeiten werden zu einer Gesamtzeit aufsummiert, aus der sich dann die mittlere Suchzeit und die Anzahl der Anschläge pro Minute errechnen lassen.

```
100 REM ================
110 REM
120 REM     SCHREIBTEST
130 REM
140 REM ================
150 REM
160 REM
170 DEF CHRND=INT(RND*94)+33
180 REM
190 REM
```

```
200 CALL CLEAR
210 DISPLAY AT(4,10):"SCHREIBTEST"
220 DISPLAY AT(5,10):"==========="
230 DISPLAY AT(10,3):"MIT WIEVIELEN ZEICHEN SOLL"
240 DISPLAY AT(11,3):"DER TEST DURCHGEFUEHRT"
250 DISPLAY AT(12,3):"WERDEN ?"
260 ACCEPT AT(12,15)VALIDATE(DIGIT):ANZAHL
270 IF ANZAHL<1 THEN STOP
280 REM
290 REM -----------------
300 REM
310 GES_ZEIT=0
320 RANDOMIZE
330 FOR LOOP=1 TO ANZAHL
340 CH=CHRND
350 IF (CH>=97)AND(CH<=122)THEN CH=CH-32
360 CH$=CHR$(CH)
370 CALL HCHAR(17,1,32,200)
380 DISPLAY AT(18,5):CH$
390 REM
400 REM -----------------
410 REM   ZEIT STOPPEN
420 REM -----------------
430 REM
440 FOR T=1 TO 10000
450 CALL KEY(0,Z,S)
460 IF S=0 THEN 480
470 GOTO 520
480 NEXT T
490 REM
500 REM -----------------
510 REM
520 Z$=CHR$(Z)
530 ZEIT=INT(T/3)/10
540 IF Z$=CH$ THEN DISPLAY AT(18,10):"  RICHTIG" ::
                   CALL SOUND(200,3000,3):: GOTO 570
550 DISPLAY AT(18,10):Z$;"          F A L S C H "
560 CALL SOUND(500,-3,1):: CALL SOUND(1,30000,30)
570 GES_ZEIT=GES_ZEIT+ZEIT
580 NEXT LOOP
590 REM
600 REM ------------------
610 REM
620 ZEIT_MITTEL=INT(GES_ZEIT/ANZAHL*10)/10
630 ANSCHLAEGE=INT(60/ZEIT_MITTEL)
640 CALL HCHAR(17,1,32,140)
650 DISPLAY AT(18,3):"DIE GEMITTELTE SUCHZEIT"
660 DISPLAY AT(19,3):"VON ";ZEIT_MITTEL;" SEKUNDEN"
670 DISPLAY AT(20,3):"ENTSPRICHT ";ANSCHLAEGE
680 DISPLAY AT(21,3):"ANSCHLAEGEN PRO MINUTE"
690 GOTO 210
700 REM
710 REM
720 REM *****************
730 REM
```

Abb. 2.4: Schreibtest-Programm

Die Anweisung 320 RANDOMIZE sorgt dafür, daß bei jedem Start des Programms eine andere Folge von Zeichen generiert wird.

Versuchen Sie, das Programm selbst zu modifizieren:

Die Zeitmessung mittels T wird so lange durchgeführt, bis die richtige Taste (und nicht irgendeine) gefunden wurde.

In einer anderen Variante könnten Sie das Programm so abändern, daß die richtigen und falschen Anschläge jeweils gezählt werden, und daß für die Fehler eine Strafzeit verrechnet wird.

EXPERIMENTE MIT DEN FARBEN DES TI-99/4A

Der TI-99/4A verfügt über die Möglichkeit, 16 Farben darzustellen, nämlich

1 transparent
2 schwarz
3 mittelgrün
4 hellgrün
5 dunkelblau
6 hellblau
7 dunkelrot
8 kornblumenblau
9 mittelrot
10 hellrot
11 dunkelgelb
12 hellgelb
13 dunkelgrün
14 magentarot
15 grau
16 weiß

Im TI-BASIC gibt es zwei Unterprogramme, die farbige Ausgaben auf dem Bildschirm erzeugen:

SCREEN und COLOR

Im Extended BASIC wurden die Möglichkeiten der Grafik und deren Farbgestaltung noch um Sprites und deren Manipulation erweitert.

Mit CALL SCREEN (F) kann die Bildschirmfarbe bestimmt werden, wobei F einer der obigen Farbcodes ist. Das Programm

```
100 FOR F = 1 TO 16
110 CALL SCREEN (F)
120 NEXT F
```

zeigt alle Farben an, die mit dem TI-99/4A erzeugt werden können. Allerdings geschieht dies viel zu schnell, um alle Farben wahrnehmen zu können. Wir bauen uns deshalb eine Zeitverzögerung nach einem der bereits bekannten Verfahren ein:

```
100 CALL CLEAR
110 FOR F = 1 TO 16
120 CALL SCREEN (F)
130 CALL SOUND (2000, 30000, 30)
140 CALL SOUND (1, 30000, 30)
150 NEXT F
```

Es werden nacheinander alle 16 Farbmöglichkeiten für die Dauer von jeweils ca. 2 sec angezeigt. Anstelle der Zeilen 130 und 140 können wir auch schreiben

```
130 CALL KEY (0, Z, S)
140 IF S = 0 THEN 130
```

und bestimmen durch Drücken irgendeiner Taste, wann die nächste Farbe erscheinen soll.

In diesen beiden Programmen werden die Farben der Reihe nach angezeigt. Das folgende Programm gestattet das Einlesen einer Zahl für den Farbcode und zeigt die entsprechende Farbe:

```
100 INPUT F
110 CALL CLEAR
```

```
120 CALL SCREEN (F)
130 PRINT "FARB CODE = "; F
140 GOTO 100
```

Dieses Programm stellt eine Endlosschleife dar, die nur durch CLEAR oder einen unzulässigen Wert für F (also F < = 0 oder F > 17) abgebrochen werden kann.

Nachdem wir wissen, wie mit SCREEN die Farbe des ganzen Bildschirms gesteuert wird, wollen wir nun einige Farbtupfer auf den Bildschirm setzen.

Wir erzeugen mit der Prozedur CHAR Zeichen, mit denen wir die Farbgestaltung ausprobieren wollen. Dazu greifen wir auf das Programm aus dem ersten Abschnitt zurück:

```
100 A$ = "FFFFC3C3C3C3FFFF"
110 B$ = "00003C3C3C3C0000"
120 CALL CHAR (130, A$)
130 CALL CHAR (131, B$)
140 CALL CLEAR
150 CALL HCHAR (10, 10, 130, 1)
160 CALL SOUND (300, 33333, 30)
170 CALL SOUND (1, 33333, 30)
180 CALL HCHAR (10, 10, 131, 1)
190 CALL SOUND (300, 33333, 30)
200 CALL SOUND (1, 33333, 30)
210 GOTO 150
```

Die neuen Zeichen mit den Codenummern 130 und 131 wollen wir jetzt farbig gestalten. Dazu dient das Unterprogramm COLOR, mit dem für eine Zeichengruppe die Vorder- und die Hintergrundfarbe festgelegt werden kann. Alle Punkte, denen in der Patterndefinition eine 1 zugeordnet wurde, werden in der Vordergrundfarbe dargestellt, die Punkte mit einer 0 in der Hintergrundfarbe. Die Festlegungen in der COLOR-Routine beziehen sich immer nur auf eine Zeichengruppe, die im TI-BASIC und im Extended BASIC unterschiedlich festgelegt sind. Diese umfassen die folgenden ASCII-Codes:

Zeichengruppe		ASCII-Codes
TI-BASIC	Ext. BASIC	
	0	30 – 31
1	1	32 – 39
2	2	40 – 47
3	3	48 – 55
4	4	56 – 63
5	5	64 – 71
6	6	72 – 79
7	7	78 – 87
8	8	88 – 95
9	9	96 – 103
10	10	104 – 111
11	11	112 – 119
12	12	120 – 127
13	13	128 – 135
14	14	136 – 143
15		144 – 151
16		152 – 159

Abb. 2.5: ASCII-Codes für die Festlegung der Zeichen in der COLOR-Routine

Der Aufruf CALL COLOR (ZG, VF, HF) bewirkt, daß alle Zeichen, die zu der Zeichengruppe ZG gehören, mit der Vordergrundfarbe VF und der Hintergrundfarbe HF dargestellt werden. Für unsere neuen Zeichen mit den ASCII-Codes 130 und 132 ist 13 die Nummer der Zeichengruppe. Die beiden Farben wollen wir durch eine Eingabe-Anweisung einlesen, um so verschiedene Farbkombinationen ausprobieren zu können.

```
100 A$ = "FFFFC3C3C3C3FFFF"
110 B$ = "00003C3C3C3C0000"
120 CALL CHAR (130, A$)
130 CALL CHAR (131, B$)
140 CALL CLEAR
```

```
150 PRINT "FARBE FUER SCREEN";
160 INPUT SF
170 PRINT "VORDER- U. HINTERGRUNDFARBE";
180 INPUT VF, HF
190 CALL SCREEN (SF)
200 CALL COLOR (13, VF, HF)
210 CALL HCHAR (10, 10, 130, 1)
220 CALL SOUND (300, 33333, 30)
230 CALL SOUND (1, 33333, 30)
240 CALL COLOR (13, HF, VF)
250 CALL HCHAR (10, 10, 131, 1)
260 CALL SOUND (300, 33333, 30)
270 CALL SOUND (1, 33333, 30)
280 CALL KEY, (0, Z, S)
290 IF S = 0 THEN 210
300 CALL SCREEN (16)
310 GOTO 140
```

Die beiden INPUT-Anweisungen gestatten es, die Farben für SCREEN und Vorder- und Hintergrund beliebig festzusetzen. Die Anweisungen in den Zeilen 280 und 290 sorgen dafür, daß durch den Sprung nach Zeile 210 die gewählten Farbkombinationen so lange angezeigt werden, bis durch Drücken einer beliebigen Taste die Anweisungen 300 und 310 ausgeführt werden. CALL SCREEN (16) bewirkt, daß der Bildschirm weiß wird, damit die vor den INPUT-Anweisungen ausgedruckten Fragen auch lesbar sind. Falls nämlich eine dunkle Screen-Farbe gewählt wird, sind die Texte nur schwer oder gar nicht zu sehen.

Farbmischung

Durch Definition eines "Schachbrettes" liegen die beiden Farben für Vorder- und Hintergrund sehr dicht beieinander, so daß die einzelnen Farben zu einer Mischfarbe verschmelzen. Wir zeichnen drei Balken auf den Schirm, je einen in der Vorder- bzw. Hintergrundfarbe und dazwischen einen Balken in der Mischfarbe.

```
100 C28$ = "FFFFFFFFFFFFFFFF"
110 C29$ = "0000000000000000"
```

```
120 C30$ = "AA55AA55AA55AA55"
130 CALL CHAR (128, C28$)
140 CALL CHAR (129, C29$)
150 CALL CHAR (130, C30$)
160 PRINT "VORDER- U. HINTERGRUNDFARBE";
170 INPUT VF, HF
180 CALL CLEAR
190 CALL SCREEN (2)
200 CALL COLOR (13, VF, HF)
210 CALL HCHAR (10, 1, 128, 32)
220 CALL HCHAR (12, 2, 130, 30)
230 CALL HCHAR (14, 1, 129, 32)
240 CALL KEY (0, Z, S)
250 IF S = 0 THEN 240
260 CALL SCREEN (16)
270 GOTO 160
```

Bei dem obigen Programm haben Sie jedesmal selbst zwei Farben vorgegeben, deren Mischung Sie sich ansehen wollen. Um alle möglichen Kombinationen anzuschauen, ändern wir das Programm so ab, daß an Stelle der INPUT-Anweisungen das Programm der Reihe nach die Farben bestimmt:

```
100 C28$ = "FFFFFFFFFFFFFFFF"
110 C29$ = "0000000000000000"
120 C30$ = "AA55AA55AA55AA55"
130 CALL CHAR (128, C28$)
140 CALL CHAR (129, C29$)
150 CALL CHAR (130, C30$)
160 FOR VF = 3 TO 13
170 HFA = VF + 1
180 FOR HF = HFA TO 14
190 CALL CLEAR
200 CALL SCREEN (2)
210 CALL COLOR (13, VF, HF)
220 FOR I = 3 TO 8
230 CALL COLOR (I, 16, 2)
240 NEXT I
250 CALL HCHAR (7, 1, 128, 32)
```

```
260 CALL HCHAR (14, 2, 130, 30)
270 CALL HCHAR (21, 1, 129, 32)
280 DISPLAY AT (9, 10) : "VF = "; VF
290 DISPLAY AT (19, 10) : "HF = "; HF
300 CALL KEY (0, Z, S)
310 IF S = 0 THEN 300
320 NEXT HF
330 NEXT VF
```

Jedesmal, wenn Sie irgendeine Taste drücken, erscheint auf dem Bildschirm die nächste Farbkombination.

Farbiger Text

Das Unterprogramm COLOR gestattet es, für jede Zeichengruppe eine Vorder- und Hintergrundfarbe zu wählen. Um zum Beispiel die Buchstaben und Ziffern in verschiedenen Farben darzustellen, müssen die Zeichengruppen 5, 6, 7 und 8, die die Großbuchstaben enthalten, und die Zeichengruppen 3 und 4, die die Ziffern enthalten, durch COLOR-Aufrufe auf die gewünschten Farben eingestellt werden.

```
100 PRINT "VF U. HF FUER BUCHSTABEN"
110 INPUT VFBU, HFBU
120 PRINT "VF U. HF FUER ZIFFERN"
130 INPUT VFZI, HFZI
140 FOR ZG = 5 TO 8
150 CALL COLOR (ZG, VFBU, HFBU)
160 NEXT ZG
170 CALL COLOR (3, VFZI, HFZI)
180 CALL COLOR (4, VFZI, HFZI)
190 INPUT L$
200 GOTO 190
```

Sie können jetzt beliebigen Text auf den Bildschirm schreiben, bei dem Buchstaben und Ziffern entsprechend den gewählten Farben erscheinen. Die Zeichen, die nicht zu den Teilzeichensätzen 3 bis 8 gehören, bleiben in der Standarddarstellung schwarz auf transparent. Wegen der Endlosschleife kann das Programm nur durch CLEAR abgebrochen werden.

DER SOUND DES TI-99/4A

Das Unterprogramm SOUND dient zur Erzeugung von Tönen und Geräuschen, die durch Dauer, Frequenz und Lautstärke festgelegt werden. Bei dem Verzögerungsprogramm haben wir bereits von der Möglichkeit, die Dauer eines Tones zu bestimmen, Gebrauch gemacht. Der Aufruf erfolgt durch

```
CALL SOUND (DAUER, FREQUENZ, LAUTSTÄRKE)
```

Die Dauer wird in Einheiten von 1 bis 4250 angegeben, die ungefähr Millisekunden (1 msec = 1/1000 sec) entsprechen, d.h. also, daß Töne und Geräusche von ca. 1/1000 sec bis ca. 4,25 sec erzeugt werden können. Während ein Ton abgespielt wird, fährt der Rechner in der Programmausführung fort. Erfolgt während des Abspielens eines Tones ein weiterer Aufruf des SOUND-Unterprogramms, so wartet der Rechner so lange, bis der erste Ton fertig gespielt ist, und startet danach den zweiten Aufruf. Wird der Wert für die Dauer negativ angegeben (also z.B. −500), dann wird der ggf. noch ausgegebene Ton abgebrochen und sofort mit der Ausgabe des neuen Tones begonnen.

Die Frequenzen werden in Hertz (1 Hz = 1 Schwingung pro Sekunde) angegeben. Erlaubt sind hierfür Werte von 110 bis 44733. Als Frequenzparameter können auch die Zahlen von −1 bis −8 angegeben werden, durch die Geräusche erzeugt werden.

Für die Lautstärke sind Werte von 0 bis 30 erlaubt, mit 0 für die größte und 30 für die geringste Lautstärke. Als erstes probieren wir das Repertoire aus. Zunächst die Geräusche:

```
100 FOR F = −1 TO −8 STEP −1
110 CALL SOUND (500, F, 2)
120 NEXT F
```

oder in der anderen Reihenfolge

```
100 FOR F = −8 TO −1
110 CALL SOUND (500, F, 2)
120 NEXT F
```

Die FOR-Schleife arbeitet im Normalfall den Schleifenparameter in aufsteigender Folge einen nach dem anderen ab. Da die Werte hier in der Reihenfolge −1, −2, ... bis −8, also in absteigender Folge abgearbeitet werden sollen, muß die Schrittweite mit Step −1 angegeben werden. Falls man dies nicht macht, führt die erste Überprüfung auf Schleifenende (−1 ist größer als −8) gleich auf das Endekriterium, und es wird sofort zur nächsten Anweisung weitergegangen.

Im SOUND-Aufruf können Sie durch Ändern der Werte für Dauer und Lautstärke deren Wirkung ausprobieren.

Als nächstes probieren wir die Töne aus:

```
100 FOR F = 110 TO 2010 STEP 100
110 CALL SOUND (200, F, 2)
120 NEXT F
```

Erzeugen einer Tonleiter

In der temperierten Tonskala wird eine Oktave in 12 Halbtöne aufgeteilt. Die Frequenzen zweier Töne, die eine Oktave auseinander liegen, unterscheiden sich um den Faktor 2. Da der Frequenzgang des Ohres logarithmisch ist, müssen die Frequenzen für die 12 Halbtöne in einer Oktave so bestimmt werden, daß sich der jeweils folgende Ton durch Multiplikation mit einem konstanten Faktor ergibt. Dieser Faktor muß so bestimmt werden, daß sich für den Oktavenabstand insgesamt der Faktor 2 ergibt. Das Ganze sei an der C-Dur-Tonleiter dargestellt:

c' d' e' f' g' a' h' c"

Die zwölf Töne in der Oktave von c' bis c" sind:

c' cis' d' dis' e' f' fis' g' gis' a' ais' h' c"

Es muß also f(c") = 2 ∗ f(c') sein, wobei f(x) die Frequenz des betreffenden Tones bezeichnet.

Ferner:

f(cis') = q ∗ f(c')

$$f(d') = q * f(cis') = q * q * f(c') = q^2 * f(c')$$

.

.

.

$$f(c'') = q * f(h') = q^{12} * f(c')$$

Damit können wir den Faktor bestimmen, es muß $q^{12} = 2$ sein, also

$$q = \sqrt[12]{2} = 2^{1/12} = 2\hat{}(1/12)$$

Für die temperierte Tonskala ist a' (Kammerton a) auf 440 Hz festgelegt.

Damit können wir vier Oktaven der temperierten Tonskala wie folgt vorspielen:

```
100 DELTA = 2^(1/12)
110 F = 110
120 FOR I = 1 TO 49
130 CALL SOUND (200, F, 2)
140 F = F * DELTA
150 NEXT I
```

Wenn wir nach Zeile 120 noch

```
125 PRINT INT (F)
```

einschieben, erhalten wir die Frequenztabelle, die in der Bedienungsanleitung abgedruckt ist.

Mit Hilfe der Frequenztabelle wollen wir jetzt ein Programm schreiben, mit dem wir einfache Melodien spielen können. Das Problem dabei ist die Eingabe der Noten. Wir müssen eine Verschlüsselung finden, mit der wir den Notenwert und die Dauer dem Programm mitteilen können. Dafür bietet es sich unmittelbar an, die Buchstaben der Notennamen zu verwenden, also C, D, E usw. Ausgehend von einem Tempo (z.B. der Dauer einer Viertelnote), werden die Längen der Töne in diesen Einheiten festgelegt. Wenn keine weitere Angabe erfolgt, ist der Ton eine Einheit lang. Durch Zahlen nach der Tonbezeichnung wird die

Dauer um den entsprechenden Faktor verlängert, es bezeichnet somit C den Ton c als Viertelnote und G2 das g als Halbnote usw. In dem nachfolgenden Programm sind in der DATA-Anweisung die Noten für „Alle meine Entchen“ codiert. Die Anweisungen in den Zeilen 580 bis 650 ordnen dem Parameter F die der Note entsprechende Frequenz zu.

Die 0 (Null) als letzte Eintragung in der DATA-Liste dient als Endekriterium. Sie verhindert, daß das Programm mit der Fehlermeldung „DATA ERROR IN 500“ beendet wird.

```
100 DATA, C, D, E, F, G2, G2, A, A, A, A,
          G4, A, A, A, A, G4, F, F, F, F, E2, E2,
          D, D, D, D, C4, 0
500 READ N$
510 IF N$ = "0" THEN STOP
520 D = 200
530 T$ = SEG$ (N$, 1, 1)
540 IF LEN (N$) = 1 THEN 210
550 DD = VAL (SEG$ (N$, 2, 1))
560 D = D * DD
570 F = 30000
580 IF T$ = "C" THEN F = 262
590 IF T$ = "D" THEN F = 294
600 IF T$ = "E" THEN F = 330
610 IF T$ = "F" THEN F = 349
620 IF T$ = "G" THEN F = 392
630 IF T$ = "A" THEN F = 440
640 IF T$ = "H" THEN F = 494
650 IF T$ = "X" THEN F = 523
660 CALL SOUND (D, F, 3)
670 CALL SOUND (1, 30000, 30)
680 GOTO 140
```

Mit diesem einfachen Codierungsschema lassen sich natürlich nur entsprechend einfache Melodien spielen. Wir wollen deshalb das Verfahren erweitern. Die Bezeichnung der Noten nur mit einem Buchstaben ist nicht eindeutig, wenn sich die Melodie über mehrere Oktaven erstreckt. Hier bedienen wir uns der üblichen Schreibweise mittels Strich, wir schreiben also E' oder C" usw. Zur Vereinfachung führen wir einen „Oktavenschalter“ ein, der durch den Buchstaben O und

einen nachfolgenden Wert von 0, 1 oder 2 (= Anzahl der Striche) die entsprechende Oktave bezeichnet. Dieser „Oktavenschalter" verändert die Höhen aller nachfolgenden Noten so, daß sie um 0, 1 oder 2 Oktaven höher werden. Damit erspart man sich das Schreiben der Striche.

Mit den Buchstaben allein können aber nur die ganzen Töne der C-dur-Tonleiter beschrieben werden. Um auch andere Melodien spielen zu können, muß auch für die Halbtöne wie fis, ges etc. eine Codierung festgelegt werden. In Anlehnung an das Kreuz benutzen wir das Zeichen # für die Halbtöne. Es bedeutet also C# cis usw. Da in der temperierten Skala Töne wie ges und fis gleich sind, ist damit ein ausreichendes Codierungsverfahren festgelegt.

Das Verfahren des „Oktavenschalters" kann leicht verallgemeinert werden, um die Tonhöhe nicht nur in ganzen Oktavensprüngen, sondern auch in kleineren Schritten zu verändern, um also die Melodie in eine andere Tonart zu transponieren. Das Verfahren dazu ist, wie Sie aus der obigen kurzen Abhandlung über die Physik der Töne entnehmen können, sehr einfach. Durch Multiplikation der Frequenz mit dem Faktor $q = 2^{1/12}$ erhält man den nächsthöheren Ton in der temperierten Zwölftonskala. Um also eine Melodie um n Halbtöne zu transponieren, müssen alle Frequenzen n mal mit dem Faktor q, also mit q^n multipliziert werden. Wir vereinbaren deshalb noch ein Zeichen, das die Transposition der nachfolgenden Noten um die angegebene Anzahl von Halbtönen bewirkt. Mit T5 wird die nachfolgende Melodie um fünf Halbtöne nach oben transponiert, also z.B. von C-Dur nach F-Dur.

In manchen Musikstücken wird eine Pause benötigt, die wir mit Pn bezeichnen wollen, d.h., daß für n Zeitintervalle eine Pause realisiert werden soll. Mit P kann auch eine längere Pause programmiert werden.

Das alles ist in dem nachfolgenden Programm „PROGRAMMIERBARE TI-ORGEL" zusammengefaßt.

```
100 REM
110 REM
120 REM          DATA Anweisung mit Notencodierung
130 REM
140 REM
150 REM
```

```
500 REM +-----------------+
510 REM |                 |
520 REM | PROGRAMMIERBARE |
530 REM |    TI - ORGEL   |
540 REM |                 |
550 REM +-----------------+
560 REM |                 |
570 REM |    VERSION 1    |
580 REM |                 |
590 REM +-----------------+
600 REM
610 REM
620 REM
630 CALL CLEAR
640 CALL SCREEN(13)
650 FOR I=2 TO 8 :: CALL COLOR(I,16,1):: NEXT I
660 DISPLAY AT(8,7):"PROGRAMMIERBARE"
670 DISPLAY AT(10,7):"  TI - ORGEL"
680 CALL HCHAR(4,3,42,28)
690 CALL HCHAR(22,3,42,28)
700 CALL VCHAR(4,3,42,18)
710 CALL VCHAR(4,30,42,18)
720 DISPLAY AT(24,22):"(C) GPR"
730 Q=2^(1/12)
740 DSTD=250
750 OKTAVSTD=1
760 TRANSP=1
770 LSTD=2
780 REM
790 REM
800 REM
810 READ N$
820 IF N$="0" THEN STOP
830 NL=LEN(N$)
840 T$=SEG$(N$,1,1)
850 IF T$="O" THEN 1120
860 IF T$="T" THEN 1190
870 OKTAV=OKTAVSTD
880 ART$="GANZ"
890 DFAKT=0
900 F=30000
910 D=4250
920 L=30
930 ZEIGER=2
940 IF ZEIGER>NL THEN 1010
950 X$=SEG$(N$,ZEIGER,1)
960 IF X$="" THEN ART$="HALB" :: GOTO 990
970 IF X$="'" THEN OKTAV=OKTAV*2 :: GOTO 990
980 DFAKT=VAL(X$)+DFAKT*10
990 ZEIGER=ZEIGER+1
1000 GOTO 940
1010 IF ART$="GANZ" THEN CALL GANZTON(T$,F)
                    ELSE CALL HALBTON(T$,F)
1020 D=DSTD
1030 IF DFAKT<>0 THEN D=D*DFAKT
1040 F=F*OKTAV*TRANSP
1050 L=LSTD
1060 IF (F>30000)OR(F<110)THEN F=30000
1070 CALL SOUND(1,30000,30)
```

```
1080 CALL SOUND(D,F,L)
1090 GOTO 810
1100 REM
1110 REM
1120 REM  OKTAVSTD AENDERN
1130 REM====================
1140 OKTAVSTD=2^(VAL(SEG$(N$,2,1)))
1150 IF NOT((OKTAVSTD=1)OR(OKTAVSTD=2)OR(OKTAVSTD=4))
                                              THEN OKTAVSTD=1
1160 GOTO 810
1170 REM
1180 REM
1190 REM   TRANSPONIEREN
1200 REM===================
1210 TF=VAL(SEG$(N$,2,NL-1))
1220 TRANSP=Q^TF
1230 GOTO 810
1240 REM
1250 REM
1260 SUB GANZTON(T$,F)
1270 REM =============
1280 REM
1290 IF T$="C" THEN F=131 :: SUBEXIT
1300 IF T$="D" THEN F=147 :: SUBEXIT
1310 IF T$="E" THEN F=165 :: SUBEXIT
1320 IF T$="F" THEN F=175 :: SUBEXIT
1330 IF T$="G" THEN F=196 :: SUBEXIT
1340 IF T$="A" THEN F=220 :: SUBEXIT
1350 IF T$="H" THEN F=247 :: SUBEXIT
1360 IF T$="P" THEN F=30000
1370 SUBEND
1380 SUB HALBTON(T$,F)
1390 REM =============
1400 REM
1410 IF T$="C" THEN F=139 :: SUBEXIT
1420 IF T$="D" THEN F=156 :: SUBEXIT
1430 IF T$="F" THEN F=185 :: SUBEXIT
1440 IF T$="G" THEN F=208 :: SUBEXIT
1450 IF T$="A" THEN F=233 :: SUBEXIT
1460 SUBEND
```

Abb. 2.6: Programmierbare TI-Orgel (Version 1)

Mit diesem Programm ist es bereits möglich, eine ganze Reihe von Melodien auf dem TI-99/4A zu spielen. Es fehlen nur noch zwei wesentliche Dinge:

Durch Variation der Lautstärke können in einem Stück bestimmte Passagen betont oder zurückgenommen werden. Was wir also noch brauchen, ist eine Codierung für die Lautstärke, die in zwei Arten möglich sein soll: durch Angabe einer generellen Lautstärke mittels Ln

(n = 0 bis 30) und durch Angabe einer Lautstärke bei einer einzelnen Note, die dann nur für diese eine Note gilt.

Das Unterprogramm SOUND erlaubt noch weitere Parameter, nämlich insgesamt dreimal die Angabe von Frequenz und Lautstärke und zusätzlich eine Geräuschangabe aus dem Wertebereich von −1 bis −8 mit Lautstärke:

```
CALL SOUND (D, F1, L1 [, F2, L2 [, F3, L3 [, F4, L4]]])
```

Diese Töne und das zusätzliche Geräusch werden gleichzeitig mit der durch den ersten Parameter bestimmten Dauer abgespielt. Um diese Möglichkeit zu nutzen, wollen wir unser Codierverfahren erweitern.

Durch * und eine nachfolgende Zahl aus dem Wertebereich 0 bis 30 wird die Lautstärke des davorstehenden Tones bestimmt. Fehlt diese individuelle Angabe, so gilt die generell festgelegte Lautstärke.

Durch einen Punkt werden ein weiterer Ton bzw. ein weiteres Geräusch angegeben, die gleichzeitig mit dem als ersten festgelegten Ton gespielt werden sollen. Diese Angabe besteht aus der Angabe der Tonhöhe, also z.B. F oder G#, und einer eventuellen Lautstärkeangabe *n. Die Dauer des Tones wird durch die Dauer des ersten Tones bestimmt, wie es der SOUND-Aufruf erfordert. Die Geräusche werden durch die Angabe −1 bis −8 festgelegt, sie werden entsprechend in den SOUND-Aufruf übernommen und haben die dort festgelegte Bedeutung.

Mit der Bezeichnung F#2*0.A#*2.C#'*4 wird zum Beispiel der Fis-Dur-Dreiklang fis, ais, cis' gespielt. Die Dauer ist zweimal das Grundintervall, die Lautstärken sind 0 für fis, 2 für ais und 4 für cis'.

Diese einigermaßen komplizierten Regeln der Musikcodierung bedingen natürlich auch einen entsprechenden Rechenaufwand bei der Umsetzung im TI-99/4A. Bei Stücken mit hohem Tempo und damit kurzer Zeitdauer für das Grundintervall kann es deshalb zu Zeitproblemen kommen, die sich in Unregelmäßigkeiten im Ablauf des Spieles äußern. Um dies zu vermeiden, haben wir den Ablauf in zwei Phasen aufgeteilt: in eine „Lernphase" und in eine „Ausführungsphase". In der Lernphase wird die Musikcodierung gelesen und entschlüsselt, und die so aufbereiteten Daten werden in eine schneller verarbeitbare Form gebracht und abgespeichert. In der Ausführungsphase werden die vorbereiteten Daten der Reihe nach abgespielt.

Erläuterungen zu dem Programm „Programmierbare TI-Orgel"

Mit DIM DS (70), L (4, 70), F (4, 70) wird der Speicher für bis zu 70 Aufrufe der SOUND-Routine festgelegt (Zeile 630 in der Version 2). In DS wird für jeden Aufruf die Dauer abgespeichert, die Felder L und F enthalten für jeden Aufruf bis zu vier Werte für die Lautstärke und die Frequenz bzw. den Geräuschcode.

Die Entschlüsselung der Musikcodierung erfolgt in den Zeilen 830 bis 1290. In N$ wird zunächst die Zeichenkette eingelesen. Sie kann direkt Noten enthalten oder „Steueranweisungen", mit denen Lautstärke, die Dauer des Grundintervalls, der Oktavenschalter etc. verändert werden können. Mit der Zeigervariablen P (wie pointer) wird jeweils auf das zu untersuchende Zeichen verwiesen, das mittels SEG$ aus der Zeichenkette in T$ herauskopiert wird (Zeile 910). In einer Reihe von IF-Abfragen wird zunächst geprüft, ob es sich um eines der Steuerzeichen handelt. Wenn ja, dann wird dadurch das Einlesen des zugehörigen Wertes veranlaßt und der betreffende Parameter auf diesen neuen Wert gesetzt. Wenn keine der IF-Abfragen wahr ist, bleibt als letztes nur die Möglichkeit, daß es sich um einen Ton handelt, der in den Zeilen 1000 bis 1150 bestimmt wird. Wenn die gesamte Zeichenkette untersucht worden ist, werden ggf. die Dauer-, Frequenz- und Lautwerte in den Feldern DS, F und L abgespeichert. Dies ist allerdings nur dann nötig, wenn in dem String tatsächlich Noten enthalten waren. Mit der Variablen MSP$ wird dies bei der Entschlüsselung festgehalten. Nachdem alle Strings mit Musikcodierung untersucht wurden, bzw. die Speicher voll sind, wird die Melodie in den Zeilen 1340 bis 1520 abgespielt.

Die Unterprogramme GT und HT (ab Zeile 1560 bzw. 1670) erklären sich selbst. Sie dienen der Zuordnung der Frequenzen für die ganzen und die halben Töne. Das Unterprogramm WERT (ab Zeile 1760) dient dazu, eine Ziffernfolge einzulesen und den zugehörenden numerischen Wert zu ermitteln. Die Zahl 234, bestehend aus der Folge der Ziffernzeichen „2", „3" und „4", hat ja die Bedeutung $2 * 100 + 3 * 10 + 4 * 1$, oder etwas anders geschrieben $(2 * 10 + 3) * 10 + 4$. Diese letzte Darstellung liegt auch der WERT-Routine zugrunde: Es wird die erste Ziffer mit 10 multipliziert, dann die nächste dazuaddiert, und dieser Vorgang wird wiederholt, bis alle Ziffern gelesen sind. Als Startwert dafür dient eine „führende Null". Es wird also im obigen Beispiel eigentlich die Ziffernfolge 0234 bestimmt. Der Vorteil eines solchen Unterprogramms besteht darin, daß damit beliebig lange Ziffernfolgen,

deren genaue Länge innerhalb einer Zeichenkette zunächst nicht bekannt ist, gelesen und ihre numerischen Werte festgestellt werden können.

```
100 REM
110 REM
120 REM      DATA Anweisung mit Notencodierung
130 REM
140 REM
150 REM
500 REM +-----------------+
510 REM |                 |
520 REM | PROGRAMMIERBARE |
530 REM |    TI - ORGEL   |
540 REM |                 |
550 REM +-----------------+
560 REM |                 |
570 REM |    VERSION 2    |
580 REM |                 |
590 REM +-----------------+
600 REM
610 REM
620 REM
630 DIM DS(70),L(4,70),F(4,70)
640 CALL CLEAR
650 CALL SCREEN(13)
660 FOR I=1 TO 8 :: CALL COLOR(I,16,1):: NEXT I
670 DISPLAY AT(8,7):"PROGRAMMIERBARE"
680 DISPLAY AT(10,7):"  TI - ORGEL"
690 CALL HCHAR(4,3,42,28)
700 CALL HCHAR(22,3,42,28)
710 CALL VCHAR(4,3,42,18)
720 CALL VCHAR(4,30,42,18)
730 DISPLAY AT(24,22):"(C) GPR"
740 Q=2^(1/12)
750 DSTD=250
760 OKTSTD=1
770 TRANSP=1
780 LSTD=2
790 C=1
800 REM
810 REM
820 REM
830 READ N$
840 DISPLAY AT(15,3):RPT$(" ",16);
850 DISPLAY AT(15,3):N$;
860 IF N$="0" THEN 1330
870 MK=1 :: P=1 :: MSP$="F" :: PD=0
880 FOR I=1 TO 4 :: F(I,C)=0 :: L(I,C)=LSTD :: NEXT I
890 D=DSTD :: OS=OKTSTD :: DS(C)=DSTD
900 LN=LEN(N$)
910 T$=SEG$(N$,P,1)
920 IF T$="L" THEN CALL WERT(N$,P,LSTD):: GOTO 1160
930 IF T$="P" THEN CALL WERT(N$,P,PD):: F(MK,C)=33333 ::
                   L(MK,C)=30 :: MSP$="T" :: GOTO 1160
```

```
940 IF T$="T" THEN CALL WERT(N$,P,TN):: TRANS=Q^TN
                                      :: GOTO 1160
950 IF T$="O" THEN CALL OKT(N$,P,OS) :: OKSTD=OS :: GOTO 1160
960 IF T$="Z" THEN CALL WERT(N$,P,DSTD):: GOTO 1160
970 IF T$="-" THEN P=P+1 :: F(MK,C)=-VAL(SEG$(N$,P,1)) ::
                             MSP$="T" :: GOTO 1160
980 IF T$="*" THEN CALL WERT(N$,P,L(MK,C)):: GOTO 1160
990 IF T$<>"." THEN 1020
1000 MK=MK+1
1010 GOTO 1160
1020 MSP$="T"
1030 IF P=LN THEN CALL GT(T$,F(MK,C)):: GOTO 1160
1040 U$=SEG$(N$,P+1,1)
1050 IF U$="" THEN CALL HT(T$,F(MK,C)):: P=P+1
          ELSE CALL GT(T$,F(MK,C))
1060 U$=SEG$(N$,P+1,1)
1070 IF U$<>"'" THEN 1110
1080 F(MK,C)=F(MK,C)*2
1090 P=P+1
1100 GOTO 1060
1110 DF=0
1120 U$=SEG$(N$,P+1,1)
1130 IF (U$>="0")AND(U$<="9") THEN DF=DF*10+VAL(U$) ::
                                 P=P+1 :: GOTO 1120
1140 D=DSTD
1150 IF (MK=1)AND(DF<>0) THEN DS(C)=D*DF
1160 P=P+1
1170 IF P<=LN THEN 910
1180 IF MSP$="F" THEN 830
1190 REM
1200 IF (F(1,C)>=30000)AND(PD<>0) THEN D=D*PD
1210 FOR I=1 TO MK
1220 FI=F(I,C)
1230 IF (FI>=110)AND(FI<30000) THEN FI=FI*OS*TRANSP ::
                                  F(I,C)=FI
1240 NEXT I
1250 REM
1260 C=C+1
1270 DISPLAY AT(18,4):"C = ";C;
1280 REM ACCEPT AT(1,1):DUMMY$
1290 GOTO 830
1300 REM
1310 REM==================
1320 REM
1330 C=C-1
1340 FOR I=1 TO C
1350 D=DS(I)
1360 F1=F(1,I)
1370 L1=L(1,I)
1380 IF F(2,I)=0 THEN CALL SOUND(D,F1,L1):: GOTO 1480
1390 F2=F(2,I)
1400 L2=L(2,I)
1410 IF F(3,I)=0 THEN CALL SOUND(D,F1,L1,F2,L2):: GOTO 1480
1420 F3=F(3,I)
1430 L3=L(3,I)
1440 IF F(4,I)=0 THEN CALL SOUND(D,F1,L1,F2,L2,F3,L3) ::
                  GOTO 1480
1450 F4=F(4,I)
1460 L4=L(4,I)
```

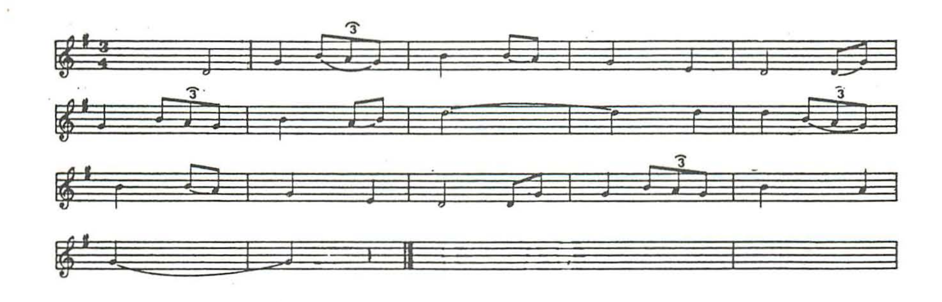
```
1470 CALL SOUND(D,F1,L1,F2,L2,F3,L3,F4,L4)
1480 NEXT I
1490 DISPLAY AT(18,4):"NOCH EINMAL        ";
1500 DISPLAY AT(19,4):"SPIELEN (J/N)      ";
1510 ACCEPT AT(19,18):JN$
1520 IF JN$="J" THEN 1340
1530 STOP
1540 REM
1550 REM
1560 SUB GT(T$,F)
1570 REM ========
1580 REM
1590 IF T$="C" THEN F=131 :: SUBEXIT
1600 IF T$="D" THEN F=147 :: SUBEXIT
1610 IF T$="E" THEN F=165 :: SUBEXIT
1620 IF T$="F" THEN F=175 :: SUBEXIT
1630 IF T$="G" THEN F=196 :: SUBEXIT
1640 IF T$="A" THEN F=220 :: SUBEXIT
1650 IF T$="H" THEN F=247
1660 SUBEND
1670 SUB HT(T$,F)
1680 REM ========
1690 REM
1700 IF T$="C" THEN F=139 :: SUBEXIT
1710 IF T$="D" THEN F=156 :: SUBEXIT
1720 IF T$="F" THEN F=185 :: SUBEXIT
1730 IF T$="G" THEN F=208 :: SUBEXIT
1740 IF T$="A" THEN F=233
1750 SUBEND
1760 SUB WERT(N$,P,W)
1770 REM ============
1780 REM
1790 W=0
1800 X$=SEG$(N$,P+1,1)
1810 IF (X$>="0")AND(X$<="9") THEN W=W*10+VAL(X$) ::
                                   P=P+1 :: GOTO 1800
1820 SUBEND
1830 REM
1840 REM
1850 SUB OKT(N$,P,W)
1860 REM ===========
1870 REM
1880 W=2^(VAL(SEG$(N$,P+1,1)))
1890 P=P+1
1900 IF (W=1)OR(W=2)OR(W=4)THEN SUBEXIT
1910 W=1
1920 SUBEND
1930 REM
1940 REM
1950 REM*******************
```

Abb. 2.7: Programmierbare TI-Orgel (Version 2)

Die Eingabe einer Melodie erfolgt wie in den vorhergehenden Beispielen in DATA-Anweisungen. Das folgende Beispiel zeigt die Noten und Codierung für das Lied „Amazing grace“:

A M A Z I N G   G R A C E

Noten und Codierung für TI-ORGEL




```
100 DATA O1,Z200,D5
105 DATA G4,Z150,H,A,G,Z200
110 DATA H4,H,A
120 DATA G4,E2
130 DATA D4,D,G
140 DATA G4,Z150,H,A,G,Z200
150 DATA H4,A,H
160 DATA D'10
170 REM DATA D'4,D'2
180 DATA D'4,Z150,H,A,G,Z200
190 DATA H4,H,A
200 DATA G4,E2
210 DATA D4,D,G
220 DATA G4,Z150,H,A,G,Z200
230 DATA H4,A2
240 DATA G6
250 DATA G6
255 DATA P10
260 DATA 0
270 REM
280 REM
290 REM
```

Abb. 2.8: Noten und Codierung mit DATA

Kapitel 3

Lernen

Nachdem uns nun eine Vielzahl von Teilprogrammen zum Bereich Spielen bekannt ist, wollen wir einige Begriffe und Daten aus der Mikrocomputertechnik diskutieren und mit Hilfe einiger Programme anwenden. Insbesondere geht es hier um einige Grundbegriffe über den Aufbau von Mikrocomputern – speziell über den TI-99/4A, über Programmierung generell und einige weitere Details zum TI-BASIC.

BITS, BYTES UND DEREN ANWENDUNG

In diesem Abschnitt sollen all denen, die noch keine Gelegenheit hatten, in die Begriffswelt der Mikrocomputer vorzustoßen, einige Erläuterungen gegeben werden.

Bit

Der grundlegende Begriff, den wir kennen sollten, ist der Begriff Bit – eine Abkürzung für binary digit –, was soviel heißt wie Binärziffer. Eine Binärziffer kann, wie die Vorsilbe "bi" bereits andeutet, nur zwei Werte oder, wie die Informatiker sagen, zwei Zustände annehmen. Diese beiden Zustände können zum Beispiel die folgenden sein:

Zustand 1:	0	wahr	ja	+ 5 V	off
Zustand 2:	1	falsch	nein	0 V	on

Wenn nun aus Binärziffern innerhalb des dualen Zahlensystems eine Binärzahl gebildet werden soll, so ist das ähnlich wie im dezimalen Zahlensystem durch das Aneinanderreihen gewichteter Binärziffern möglich. Gewichten heißt in diesem Zusammenhang, daß je nach der Anordnung der einzelnen Binärziffer innerhalb der Binärzahl diese einen bestimmten Beiwert (Multiplikator) erhält.

Die Wertigkeiten werden dabei jeweils von rechts nach links wie folgt gebildet:

$$\ldots\ldots 2^3 * B3 + 2^2 * B2 + 2^1 * B1 + 2^0 * B0$$
$$= \ldots\ldots 8 * B3 + 4 * B2 + 2 * B1 + 1 * B0$$

Für die Binärzahl 1001 ergibt sich somit als Dezimalzahl

$$8 * 1 + 4 * 0 + 2 * 0 + 1 * 1$$
$$= 8 \quad + 0 \quad + 0 \quad + 1$$
$$= 9$$

Je nach der speziellen Vereinbarung wird eine bestimmte Anzahl von Binärziffern zu einer Einheit zusammengefaßt.

Soll eine Zahl als Binärzahl geschrieben oder codiert werden, so erfolgt das im allgemeinen durch ein einfaches Aneinanderreihen von Binärziffern entsprechend der Größe der Zahl. Die Umwandlung von Binärzahlen in Dezimalzahlen ist dabei besonders einfach, indem, ausgehend von der niedrigsten Wertigkeit, die entsprechenden Werte addiert werden. Das folgende einfache Programm beschäftigt sich damit.

Wir geben hierzu die Binärzahl in Form einer Zeichenkette (0 und 1) ein und fragen diese von rechts nach links ab.

```
1000 REM ====================
1010 REM  UMWANDLUNGSPROGRAMM
1020 REM  OKTAL/DEZIMAL/HEXAD
1030 REM    AUS BINAERZAHLEN
1040 REM ====================
1050 REM HAUPTROUTINE
1060 ON WARNING NEXT
1070 GOSUB 1340
1080 FOR I=1 TO 10
1090 DB=0
1100 ACCEPT AT(I+4,1)VALIDATE("01")SIZE(8):B$
1110 IF LEN(B$)=0 THEN GOTO 1100
1120 GOSUB 1230
1130 REM
1140 DISPLAY AT(I+4,18):USING "###":DB
1150 DISPLAY AT(I+4,25):H1$
1160 DISPLAY AT(I+4,26):H2$
1170 NEXT I
1180 GOSUB 1420
1190 GOTO 1050
1200 REM ********************
```

```
1210 REM UNTERPROGRAMME
1220 REM ********************
1230 REM UP1 - UMWANDLUNG
1240 REM ********************
1250 L=LEN(B$)
1260 FOR J=L TO 1 STEP -1
1270 DB=DB+2^(J-1)*VAL(SEG$(B$,J,1))
1280 H1=INT(DB/16):: H2=DB-H1*16
1290 IF H1<=9 THEN H1$=CHR$(H1+48)ELSE H1$=CHR$(H1+55)
1300 IF H2<=9 THEN H2$=CHR$(H2+48)ELSE H2$=CHR$(H2+55)
1310 NEXT J
1320 RETURN
1330 REM ********************
1340 REM UP2 - UEBERSCHRIFT
1350 REM ********************
1360 CALL CLEAR
1370 DISPLAY AT(1,1):"----------------------------"
1380 DISPLAY AT(2,1):"BINAER    OKTAL DEZIM HEXAD"
1390 DISPLAY AT(3,1):"----------------------------"
1400 RETURN
1410 REM ********************
1420 REM UP3 - STOP/WEITER
1430 REM ********************
1440 DISPLAY AT(22,1):"FORTS., DANN J EINGEBEN"
1450 ACCEPT AT(22,25)SIZE(1):W$
1460 IF W$="J" THEN RETURN
1470 CALL CLEAR
1480 REM ********************
1490 END
```

Abb. 3.1: Umwandlungsprogramm für Binärzahlen

Im allgemeinen werden Binärziffern, wie bereits angedeutet, in Paketen zu Einheiten zusammengefaßt. Die am meisten gebrauchten Einheiten sind:

- Dreier-Bit-Gruppen für die Verschlüsselung des Oktalcodes,
- das Halbbyte oder Nibble mit 4 Bit für die Verschlüsselung des Dezimal- und Hexadezimalcodes,
- das Byte mit 8 Bit für die Verschlüsselung von Zeichen-Codes.

Da es im Hexadezimalsystem 16 Ziffern gibt, werden für die Ziffern, die im Dezimalsystem den Werten 10 bis 15 entsprechen, neue Zeichen benötigt. Als Ziffernzeichen haben sich dafür die Buchstaben A bis F eingebürgert. Es ist also zu beachten, daß im Zusammenhang mit Hexadezimalzahlen die Zeichen A bis F Ziffern sind und nicht Buchstaben.

Dezimal	Binär	Oktal	Hexadez.
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
.	.	.	.
.	.	.	.

Abb. 3.2: Zahlensysteme

Byte

Das Byte nimmt eine Sonderstellung ein. Zum einen werden alle Zeichen, mit denen wir hier umgehen, mit einem Byte verschlüsselt. Dies geschieht mit Hilfe des ASCII-Codes. Darüber hinaus wird die Speicherkapazität von Rechnersystemen häufig in Kilo-Byte angegeben. Kilo steht für duales Kilo, 1 K = 1024 = 2^{10}.

Wort

Mit Wort wird die logische Informationseinheit bezeichnet, die in einem Arbeitsschritt verarbeitet werden kann. Bei 8-Bit-Mikroprozessoren ist ein Wort gleich ein Byte. Der Mikroprozessor des TI-99/4A ist der 16-Bit-Prozessor TMS 9900. Der TMS 9900 verarbeitet 16 Bit in einem Arbeitsschritt, folglich besteht ein Wort des TMS 9900 aus 16 Bits oder 2 Bytes.

DER AUFBAU DES TI-99/4A

Grundaufbau

Wie jeder Computer besteht der TI-99 aus den drei Grundeinheiten CPU, Ein-/Ausgabeeinheit und Speicher, die durch eine Reihe von Zusatzkomponenten unterstützt werden.

SPEICHER	CPU	EIN-/AUSGABE
– RAM	TMS 9900	– Tastatur
– ROM		– Bildschirm
– PROM		– Drucker
		– Kassette
		– Diskette

Im realen Aufbau werden diese Hauptkomponenten durch eine Vielzahl von Bausteinen und Leitungen miteinander verbunden. Im folgenden gehen wir kurz auf die Hauptkomponenten des TI-99 ein (vgl. Abb. 3.3).

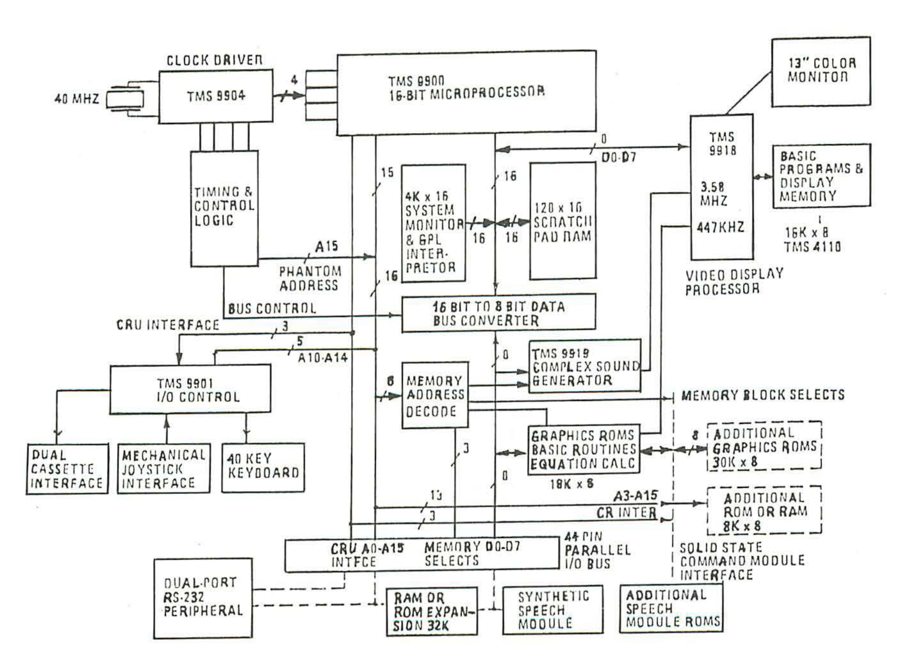
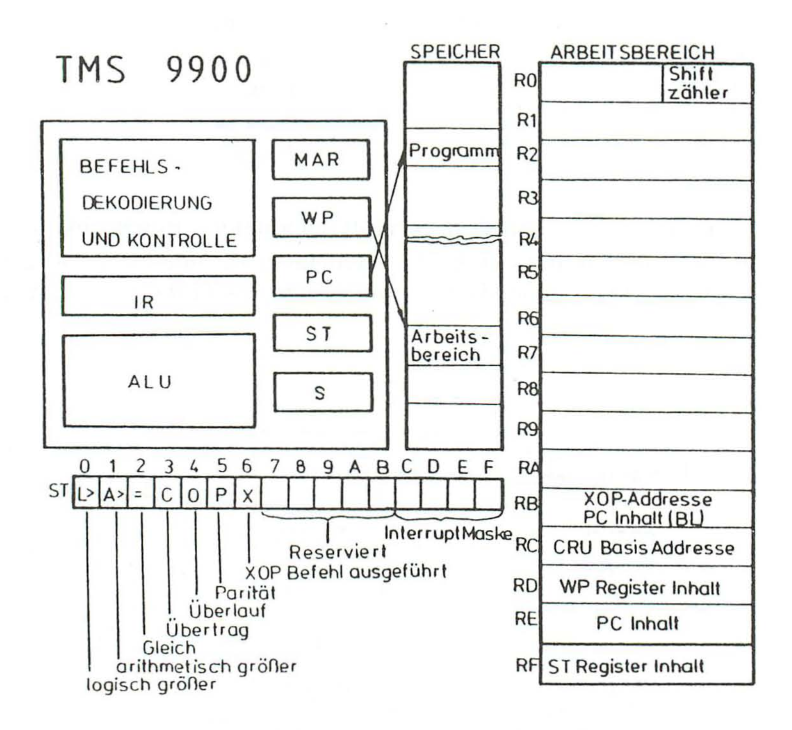


Abb. 3.3: TI-99/4A System-Blockdiagramm

CPU (TMS 9900)

Central Processing Unit. Sie bestimmt die innere Leistungsfähigkeit des Computers und ist in unserem Falle der 16-Bit-Mikroprozessor TMS 9900. Er verfügt intern über 3 Hardwareregister für die schnelle Zwischenspeicherung und über 16 Softwareregister. Sein Befehlsvorrat umfaßt 69 Grundbefehle, die entsprechend der Stellung eines Befehls-



IR	instruction register (Befehlsregister)
ALU	arithmetic and logic unit (Rechenwerk)
MAR	memory address register (Speicher-Adreßregister)
WP	workspace register (Arbeitsbereichsregister)
PC	program counter (Programmzähler)
ST	status register (Statusregister)
S	operand address register (Operanden-Adreßregister)

Abb. 3.4: Übersichtsdiagramm des TI-99/4A

zählers aus dem Programmspeicher übernommen werden. Die Verarbeitungsgeschwindigkeit wird wie bei allen Prozessoren von der Frequenz des Taktgenerators bestimmt.

Taktgenerator (TMS 9904)

Der Taktgenerator liefert die vom Prozessor geforderten Taktsignale (vier versetzte Rechtecksignale) mit einer Frequenz von 3.3 MHz, die dieser aus einer Grundfrequenz von 40 MHz ableitet. Wie wir aus dem Schaltdiagramm erkennen können, liefert der Taktgenerator auch Signale für die Steuerung der Zeit- und Kontroll-Logik.

Bus

Die Verbindung zu den anderen Bausteinen des TI-99 nimmt der TMS 9900 über drei Leitungsgruppen auf. Logisch sind dies der Daten-, der Adreß- und der Steuerbus. Über den Datenbus werden 8-Bit- oder 16-Bit-Daten zu den angeschlossenen Bausteinen übertragen, die mit den Adreßinformationen der über den Adreßbus gesendeten Signale übereinstimmen. Die Abstimmung und Zeit-Taktung geschieht dabei mit Hilfe der Steuersignale auf dem Steuerbus. Da die Anschlüsse des TMS 9900 physikalisch Mehrfachfunktionen aufweisen (z.B. sind je nach Takt bestimmte Anschlüsse einmal 8-Bit-Datenleitungen und ein anderes Mal 16-Bit-Datenleitungen), ist die logische Trennung der Anschlußfunktionen aus dem Schaltdiagramm allein nicht möglich. Vielmehr müssen die Zustände der Bus-Kontrollsignale berücksichtigt werden. Für die Umwandlung der 16-Bit-Daten des 16-Bit-Daten-Busses in 8-Bit-Daten wird im TI-99 ein Bus-Converter eingesetzt, der von den Bus-Kontrollsignalen der Zeit- und Kontroll-Logik gesteuert wird.

Zeit- und Kontroll-Logik (Timing & Control Logic)

Die Zeit- und Kontroll-Logik übernimmt neben der Steuerung des Bus-Konverters noch die Erzeugung eines zusätzlichen Adreßbits (Phantom Address), das den Adreßbus erweitert.

Ein-/Ausgabesteuerung (TMS 9901 I/O-Control)

Die Ein-/Ausgabesteuerung wird beim TI-99 mit Hilfe des Bausteins TMS 9901 realisiert, der die von der CPU gesendeten Ein-/Ausgabein-

formationen entsprechend den über die Adreßbits A10 – A14 gesendeten Informationen auf

- eine Doppel-Kassettenlaufwerk-Schnittstelle,
- die Joystick-Schnittstelle und die
- Tastatur-Schnittstelle

aufteilt.

Während die Tastatur zur Eingabe von Informationen (Steuerbefehlen, Programmen und Daten) vorgesehen ist, dient als Basisschnittstelle für die Ausgabe von Informationen ein TV-Schirm, der über ein entsprechendes Bild-/Tonsignal aus dem Baustein TMS 9918 angesteuert wird. Der TMS 9918 als Video-Display-Processor transformiert die über den Prozessor erzeugten Signale in Bild- und Toninformationen, die direkt über die Antennenbuchse eines Farb- oder Schwarzweißfernsehers empfangen werden können. Darüber hinaus empfängt er die Signale des eingebauten Tongenerators (TMS 9910) und Grafik-ROMs, die wie die anderen Bausteine ebenfalls von der CPU angesteuert werden.

Hiermit sind wir nun an einer Stelle angelangt, an der wir die Speicherbereiche des TI-99 erläutern müssen.

Alle Speicherbausteine werden über die Signale des Adreßbusses adressiert und über die Leitungen des Datenbusses ein- bzw. ausgelesen. Es sind dies RAMs und ROMs.

RAM

Mit RAM (random access memory) werden Halbleiter-Speicherbausteine bezeichnet, die auf 1024, 4096 usw. Adressen binäre (1/0 –) Informationen speichern. Diese werden auf Anweisung an speziellen Datenleitungen aus- bzw. eingelesen. Im TI-99 werden hierzu die dynamischen Speicherbausteine TMS 4116 verwendet. Diese speichern jeweils 16 KBit und bilden als Gruppe von 8-Bit den Grundspeicher mit 16K-Byte. In ihm werden alle Programme in BASIC, die dazugehörigen Daten und die für die Steuerung des TI-99 intern notwendigen Daten- und Steuerinformationen gespeichert.

ROM

ROMs (read only memory) können im Gegensatz zu den oben beschriebenen RAMs nur gelesen werden. Die in ihnen enthaltenen Informatio-

nen wurden vom Hersteller bereits bei der Erzeugung (Maskenerstellung) dort in Form einer Tabelle abgelegt. An ihren Datenausgängen werden je nach anliegender Adresse bestimmte Informationen erzeugt. So werden z.B. im internen 18K × 8K-ROM je nach anliegender Adreßinformation (z.B. bestimmter Code von der CPU) die Grafiksymbole erzeugt und vom Video-Display-Prozessor in Bildsignale umgewandelt.

PROMs, EPROMs und EEPROMs

Diese PROMs sind Abwandlungen von ROMs, deren Informationen nach dem Herstellungsprozeß des ROM noch geändert bzw. zum erstenmal erzeugt werden können. PROMs (programmable ROMs) können mit speziellen Programmiergeräten (Prommer) programmiert, d.h. mit bestimmten Informationen gefüllt werden, die dann allerdings wie bei den Masken-ROMs unlöschbar gespeichert sind.

EPROMs (erasable PROMs) können nach dem Programmieren durch UV-Licht gelöscht und danach erneut programmiert werden. Sie sind dadurch mehrfach benutzbar.

EEPROMs (electrically erasable PROMs) können auf eine noch leichtere Art (elektrisch) gelöscht werden und eignen sich daher besonders für Entwicklungsaufgaben.

Wegen der hohen Stückzahlen und der festen internen Struktur der Informationen sind die ROMs im TI-99 als Masken-ROMs ausgelegt. Im einzelnen sind dies zum Beispiel die ROMs für

- Grafikroutinen im Grafik-ROM,
- die Steuerroutinen im Monitor und die
- BASIC-Routinen im Steckmodul BASIC.

Den RAMs und ROMs sind innerhalb des Adreßraumes des TI-99 bestimmte Adreßbereiche zugeordnet, über die die in ihnen enthaltenen Routinen und Umwandlungen angesprochen werden. Kenntnisse der Speicheraufteilung (memory allocation) eröffnen uns einen Einstieg in die Struktur des TI-99/4A.

Die Speicheraufteilung zeigt uns, daß einige Ein-/Ausgabeeinheiten als sogenannte memory-mapped-devices organisiert sind, das heißt, daß sie unmittelbar über den direkt adressierbaren Speicherraum angesprochen werden können.

Hexadezimal-Adresse	Verwendung
0000 – 0FFE	für interne Zwecke
1000 – 10FE	reserviert
1100 – 11FE	Disketten-Steuereinheit
1200 – 12FE	reserviert
1300 – 13FE	RS 232 (I)
1400 – 14FE	reserviert
1500 – 15FE	RS 232 (II)
1600 – 16FE	reserviert
1700 – 17FE	reserviert
1800 – 18FE	Thermodrucker
1900 – 19FE	(besetzt für zukünftige Erweiterung)
2000 – 3FFF	(besetzt für zukünftige Erweiterung)
4000 – 5FFF	Peripherie-Erweiterung (Ein-/Ausgabe-Stecker)
6000 – 7FFF	Steckeinheit für RAM/ROM
8000 – 83FF	interner RAM-Bereich
8400 – 8700	Tonerzeugung
8800 – Video-Display-Processor (Read Data)	
8802 – Video-Display-Processor (Read Status)	
8C00 – Video-Display-Processor (Write Data)	
8C02 – Video-Display-Processor (Write Address)	
9000 – Sprache Lesen	
9400 – Sprache Schreiben	
9800 – Spielkassette (Read Data)	
9802 – Spielkassette (Read Address)	
9C00 – Spielkassette (Write Data)	
9C02 – Spielkassette (Write Address)	

Abb. 3.5: Speicheraufteilung (memory allocation)/Geräteadressierung

Externer Ein-/Ausgabebus (parallel I/O-Bus)

Der 44polige Ein-/Ausgabebus ermöglicht die eigentliche Erweiterung des TI-99. Es sind dies insbesondere

- zwei RS 232-Schnittstellen, mit denen z.B. ein Drucker und ein Plotter angesteuert werden können,
- eine Speichererweiterungs-Schnittstelle für die Erweiterung des Arbeitsspeichers um 32 KB auf insgesamt 48 KB und die
- Sprachschnittstelle für den Anschluß eines Spracherzeugungsmoduls.

PROGRAMMIERUNG

Unter Programmierung im engeren Sinn wird zunächst nur das eigentliche Schreiben eines Programmes verstanden, was oft auch als Codierung bezeichnet wird. In einem umfassenderen Sinn gehört zur Programmierung auch die Entwicklung des Lösungsverfahrens für eine gestellte Aufgabe. Alle Regeln, die zusammen die Lösung eines Problems beschreiben, werden Algorithmus genannt. Ein Programm ist dann eine Folge von Anweisungen, die vom Benutzer spezifiziert werden und die Ausführung eines bestimmten Algorithmus bewirken. Bei den Programmen unterscheiden wir im allgemeinen drei Niveaus:

- Programme in Maschinencode,
- Programme in Assembler und
- Programme in einer höheren Programmiersprache.

Programme in Maschinencode

Beim Maschinencode handelt es sich um Programme, die in einer dem Prozessor direkt verständlichen Form vorliegen. Die Codes, die hierzu verwendet werden, sind der Binär-, der Oktal- oder der Hexadezimalcode. Die Anweisungen an den Prozessor, die auf diese Weise codiert werden, sind die sogenannten Maschinenbefehle, die aus dem Arbeitsspeicher des Prozessors entnommen und in der ALU (Arithmetic and Logic Unit) abgearbeitet werden.

Eine wesentliche Eigenschaft der in Maschinencode geschriebenen Programme ist der direkte Bezug der Befehle zu den physikalischen Speicheradressen (im TMS 9900 von 0000 bis FFFF) und den internen Registern (z.B. die 3 Hardware- und 16 Softwareregister):

Arbeitsspeicheradressen

0000	erste mögliche Adresse
0001	
0002	
....	
....	
1F0A	erstes Byte des i-ten Befehls
....	
....	
FFFF	letzte mögliche Adresse

Register

PROGRAMMZÄHLER mit der Adresse des nächsten auszuführenden Befehls

ARBEITSBEREICHSZEIGER mit der Adresse des aktuellen Arbeitsspeicherbereiches

STATUSREGISTER mit dem Zustand des abgelaufenen Befehls

R0 bis R15 als Softwareregister der CPU mit freien und belegten Informationen für den Programmablauf

Um Programme im Maschinencode schreiben zu können, muß man die Art und den allgemeinen Aufbau der Maschinenbefehle kennen. Der Prozessor TMS 9900 unterscheidet neun verschiedene Befehlsarten:

1. arithmetische Grundbefehle (Addition, Subtraktion)
2. Sprungbefehle (Sprung auf eine bestimmte Adresse / Sprung über einen bestimmten Adreßbereich, unbedingter Sprung / bedingter Sprung je nach dem Wert des Statusregisters)
3. logische Befehle (AND, OR, EXOR, INVERT usw.)
4. CRU-Befehle (Befehle der Communication Register Unit als Befehle für die Steuerung der Ein-/Ausgabeeinheiten)
5. Schiebebefehle (Rechts-, Linksschieben von ALU- oder Registerinhalten)
6. Operationen
7. Kontrollbefehle
8. Datenübergabebefehle
9. Sonderbefehle (Multiplikation, Division, Unterprogrammaufruf)

Jeder Befehl hat dabei mindestens die Länge eines Wortes (hier 16 Bits = 2 Bytes). Für die Eingabe eines Programmes direkt in Maschinencode müßten freie Speicherplätze des TI-99/4A im Arbeitsspeicherbereich direkt mit den gewünschten Befehlscodes (Op-Codes) gefüllt werden. Bei entsprechender Codierung wären auf diese Weise kleine Unterprogramme in Maschinencode direkt vom BASIC aus zu übergeben, zu starten und hiermit für die Erhöhung der Verarbeitungsgeschwindigkeit heranziehbar. Leider ist es uns bisher noch nicht gelungen, vom BASIC aus einen Einstieg in den Arbeitsspeicher zu finden, so daß zum jetzigen Zeitpunkt der einfachste Einstieg in die Maschinenprogrammierung nur über den Assembler des Mini-Memory (Solid State Module) möglich ist.

Was allerdings möglich ist, ist das Auslesen des Maschinencodes vom BASIC aus. Folgendes Listing zeigt eine Routine, mit der bestimmte Speicherbereiche vom BASIC aus angezeigt werden können.

```
1000 REM ====================
1010 REM       LESEPROGRAMM
1020 REM           FUER
1030 REM       MASCHINENCODE
1040 REM AUS DEM TI-SPEICHER
1050 REM ====================
1060 REM HAUPTROUTINE
1070 ON WARNING NEXT
1080 ZZ=1
1090 GOSUB 1200
1100 GOSUB 1330
1110 FOR I=TAB1 TO TAB2
1120 IF ZZ>15 THEN GOSUB 1580 :: GOSUB 1330 :: ZZ=1
1130 GOSUB 1410
1140 GOSUB 1500
1150 ZZ=ZZ+1
1160 NEXT I
1170 GOSUB 1580
1180 GOTO 1000
1190 REM ********************
1200 REM UP1 EINLESEN WERTEBEREICH
1210 REM ********************
1220 CALL CLEAR
1230 DISPLAY AT(1,1):"----------------------------"
1240 DISPLAY AT(2,1):"L2 -  AUSL. DES TI-SPEICHERS"
1250 DISPLAY AT(3,1):"----------------------------"
1260 DISPLAY AT(5,1):"SPEICHERANFANG"
1270 ACCEPT AT(5,22)VALIDATE(DIGIT)SIZE(5):TAB1
1280 DISPLAY AT(6,1):"SPEICHERENDE"
1290 ACCEPT AT(6,22)VALIDATE(DIGIT)SIZE(5):TAB2
1300 IF TAB2>32767 OR TAB2<TAB1 THEN 1290
1310 RETURN
```

```
1320 REM *******************
1330 REM UP2 UEBERSCHRIFT TABELLE
1340 REM *******************
1350 CALL CLEAR
1360 DISPLAY AT(1,1):"----------------------------"
1370 DISPLAY AT(2,1):"ADRESSE   INH-DEZ  INH-HEX"
1380 DISPLAY AT(3,1):"----------------------------"
1390 RETURN
1400 REM *******************
1410 REM UP3 - BYTEWERT LESEN UND HEX-WERT BERECHNEN
1420 REM *******************
1430 CALL PEEK(I,M)
1440 H1=INT(M/16)
1450 H2=M-H1*16
1460 IF H1<=9 THEN H1$=CHR$(H1+48)ELSE H1$=CHR$(H1+55)
1470 IF H2<=9 THEN H2$=CHR$(H2+48)ELSE H2$=CHR$(H2+55)
1480 RETURN
1490 REM *******************
1500 REM UP4 - DRUCKZEILE
1510 REM *******************
1520 DISPLAY AT(ZZ+4,1):I
1530 DISPLAY AT(ZZ+4,10):M
1540 DISPLAY AT(ZZ+4,20):H1$
1550 DISPLAY AT(ZZ+4,21):H2$
1560 RETURN
1570 REM *******************
1580 REM UP5 - STOP/WEITER
1590 REM *******************
1600 DISPLAY AT(22,1):"FORTS., DANN J EINGEBEN"
1610 ACCEPT AT(22,25)SIZE(1):W$
1620 IF W$="J" THEN RETURN
1630 CALL CLEAR
1640 REM *******************
1650 END
```

Abb. 3.6: Leseprogramm für das Auslesen von Maschinencode

```
----------------------------
L2 -  AUSL. DSE TI-SPEICHERS
----------------------------

SPEICHERANFANG        1000
SPEICHERENDE          1100
```

```
----------------------------
ADRESSE    INH-DEZ   INH-HEX
----------------------------

  1000       131       83
  1001       199       C7
  1002         2       02
  1003         1       01
  1004        23       17
  1005       192       C0
  1006       193       C1
  1007        69       45
  1008        22       16
  1009        14       0E
  1010         2       02
  1011         1       01
  1012        23       17
  1013       144       90
  1014        10       0A

FORTS., DANN J EINGEBEN  J


----------------------------
ADRESSE    INH-DEZ   INH-HEX
----------------------------

  1015        39       27
  1016        24       18
  1017        10       0A
  1018         2       02
  1019         1       01
  1020        23       17
  1021        96       60
  1022         9       09
  1023       247       F7
  1024        24       18
  1025         6       06
  1026         2       02
  1027         1       01
  1028        23       17
  1029        48       30

FORTS., DANN J EINGEBEN  N
```

Abb. 3.7: Auszug aus dem Arbeitsspeicherbereich

Für diejenigen, die wissen wollen, welche Befehle hinter den gelisteten Operation-Codes stecken, haben wir im Anhang eine Zusammenfassung der Maschinenbefehle aufgelistet.

Programme in Assembler

Mit Assembler wird ein Programm bezeichnet, das eine symbolhafte Darstellung (mnemotechnisch) der Rechnersprache als Eingabe verarbeitet und in Maschinencode für die direkte Ausführung im Prozessor umwandelt. Hierbei wird der Quell- oder Ursprungscode (Source-Code) in den endgültigen binären Zielcode (Object-Code) übersetzt. Ein Assembler ist dabei genau wie die Maschinensprache auch maschinen- bzw. prozessorabhängig und erfordert genaue Kenntnisse der Maschinenbefehle. Ein typischer Ablauf vom Problem bis zur Ausführung des Maschinenprogramms zeigt folgende Abbildung:

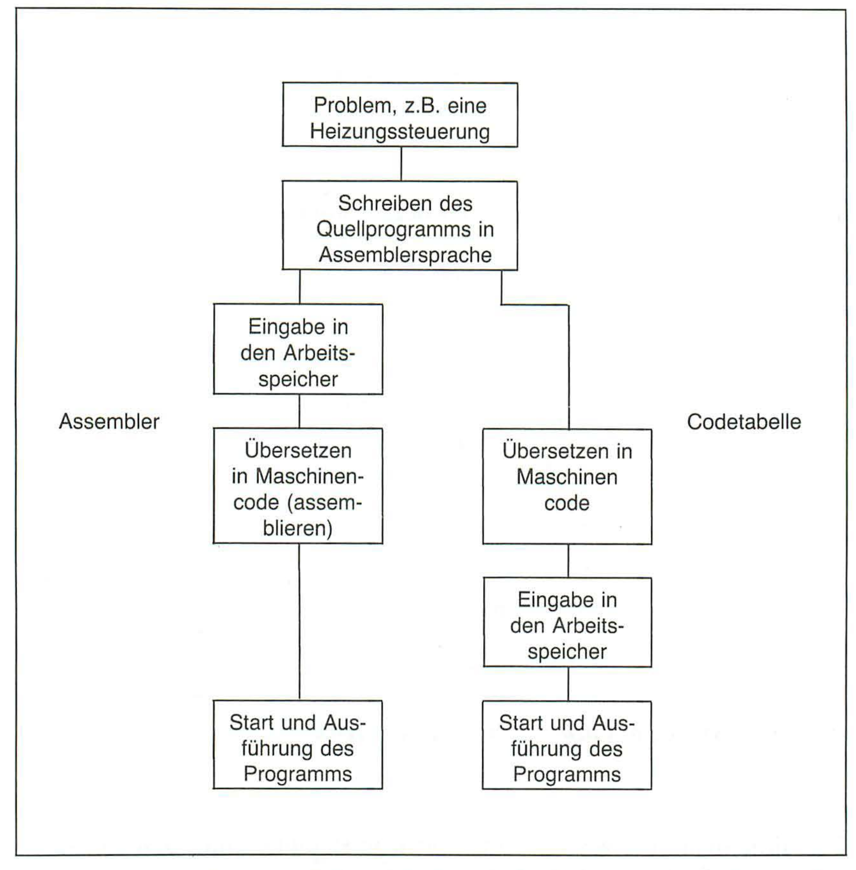


Abb. 3.8: Erstellung und Ausführung eines Maschinenprogramms über Assembler bzw. über eine Codetabelle (manuell)

Programme in höheren Programmiersprachen

Höhere Programmiersprachen (high level languages) ähneln der natürlichen (englischen) Sprache und verfügen meist über einen mächtigen Befehlsvorrat. Je nach der ursprünglichen Verwendung sind die höheren Programmiersprachen meist auf bestimmte Anwendungsbereiche zugeschnitten. Beispiele für höhere Programmiersprachen sind:

- Pascal
- FORTRAN
- BASIC
- COBOL
- APL
- ALGOL
- PL/M
- C

Auch hier werden im allgemeinen Quell- und Zielprogramme unterschieden. Je nachdem, ob die Quellprogramme direkt in den Maschinencode, in einen Zwischencode oder in einen verschiebbaren Code übertragen werden, unterscheiden wir

- Interpreter
- Semicompiler/Zwischencode-Interpreter
- Compiler

Interpreter

Interpreter (Interpretierer) sind Übersetzungsprogramme, die Anweisungen einer höheren Programmiersprache ausführen. Sie analysieren dabei jede Anweisung und führen sie direkt aus. Der Vorteil einer solchen Direktbearbeitung ist, daß, wie Sie es vom BASIC her kennen, jederzeit zusätzliche Anweisungen hinzugefügt oder andere Anweisungen gelöscht werden können.

Dieser Vorteil wird durch den Nachteil erkauft, daß die Programmausführung wegen des Interpretationsvorgangs, der zur Ausführung der eigentlichen Operation jeweils hinzukommt, relativ langsam ist. Im TI-99/4A sind das TI-BASIC und das Extended BASIC als Interpreter ausgeführt.

Semicompiler / Zwischencode-Interpreter

Semicompiler (Halbübersetzer), die vereinzelt auch Zwischencode-Interpreter genannt werden, übersetzen den Quellcode in einen Zwischencode, der dann von einem Interpreter analysiert und ausgeführt wird. Hierdurch wird eine gewisse Laufzeit- und Speicherraumverringerung erreicht. Ein weiterer Grund für den Einsatz von Semicompilern ist auch ein gewisser Schutz vor einer einfachen Änderung des lauffähigen Zwischencodeprogramms, da Änderungen nur über einen neuen Übersetzungslauf realisiert werden können. Semicompiler sind vor allem die im CP/M verfügbaren höheren Programmiersprachen CBASIC (Commercial BASIC) und Pascal/Z.

Compiler

Compiler oder Translator (Übersetzer) sind Programme, die Anweisungen einer höheren Programmiersprache in binäre Anweisungen (Objectcode) umwandeln, die direkt durch den Prozessor ausgeführt werden können. Der Übersetzungsvorgang wird dabei nur einmal bei der Umformung des Quellprogramms durchgeführt. Jede Änderung des Quellprogramms erfordert deshalb einen vollständig neuen Übersetzungslauf. Nach dem Übersetzen ist im allgemeinen noch die Durchführung eines Bindungslaufes (linken, Linker) notwendig, in dem bestimmte Basisroutinen, die im Objectcode nur als Makros oder Unterprogramme angesprochen werden, aus einer Bibliothek eingebunden und damit dem Objectprogramm zur Verfügung gestellt werden.

Wenn auch die Programmerstellung über Compiler durch die verschiedenen Arbeitsschritte (Quellcode-Erfassung, Compilieren, Linken, Ausführen) aufwendiger ist als über Interpreter, zahlt sich dies insbesondere bei Laufzeit- und Speicherraum-kritischen Programmen aus. Darüber hinaus sind die Programme gegen leichte Änderung bei Einbeziehung eigener Makros in die Bibliothek wesentlich besser geschützt als Interpreter- oder Semicompiler-Programme. Mit dem TI-99/4A können Compiler zur Zeit nicht gefahren werden.

Strukturierte Programmierung

Unter dem Begriff der strukturierten Programmierung sind eine Reihe von Techniken zusammengefaßt, die zur Erhöhung der Qualität und Verständlichkeit von Programmen entworfen wurden. Erreicht wird dies insbesondere durch eine verstärkte Disziplin der Programmierer

bei der Strukturierung der Programme. Das strukturierte Programmieren erfordert eine genaue Problemspezifikation. Im sogenannten „top-down“-Entwurf wird von der Aufgabenstellung ausgehend die Strukturierung schrittweise durch Verfeinerung des vorhergehenden Schrittes vorgenommen. Erleichtert wird diese Aufgabe durch die Verwendung von Block- bzw. Modular-strukturierten Programmen.

Die verfügbaren höheren Programmiersprachen unterstützen die strukturierte Programmierung dabei mehr oder weniger durch den Einsatz strukturierter Sprachelemente. Es sind dies Kontrollstrukturen wie

```
–    IF        ... THEN     ... ELSE
–    DO        ... WHILE
–    REPEAT    ... UNTIL
```

sowie die Möglichkeit der Bildung

- in sich abgeschlossener logischer Programm-Module,
- von Prozeduren,
- von eingeschränkten Gültigkeitsbereichen für Variablen

und damit von Elementen mit modularer Struktur. Unterstützt wird die strukturierte Programmierung vor allem von Pascal und von einigen wenigen BASIC-Versionen (z.B. Structured BASIC von Cromemco).

Modulare Programmierung

Modulare Programmierung bezeichnet die Strukturierung in sich abgeschlossener Teilprogramme (Module) mit nur einem Ein- und Ausgang, die über eine einfache Kontrollstruktur miteinander verbunden werden. Modulare Programme lassen sich auch mit dem hier verfügbaren TI-BASIC und Extended BASIC durchführen.

Kapitel 4
Arbeiten

In diesem Kapitel wollen wir Ihnen eine Reihe von Anwendungen Ihres TI-99/4A vorstellen. Nach den Vorbereitungen in den vorangegangenen Kapiteln werden wir uns in den Erläuterungen auf die wesentlichen Dinge der Programmiertechnik und des Verständnisses der Probleme beschränken.

Als erstes wird eine Reihe von grundlegenden Begriffen und die Entwicklung von Techniken und Programmteilen vorgestellt, die man immer wieder braucht und auf die in den folgenden Anwendungsprogrammen auch wieder Bezug genommen wird. Zunächst werden wir uns mit Fragen der Programmiertechnik und des Programmierstils beschäftigen. Dazu gehören die Gestaltung von Ein- und Ausgabe, die Kontrolle der eingegebenen Daten sowie die Gestaltung der Eingabe mittels Menütechnik. In vielen Anwendungen spielen Datumsangaben und deren Weiterverarbeitung eine Rolle. Wir entwickeln uns dafür einige Unterprogramme, die für kalenderbezogene Arbeiten benötigt werden.

Es folgen dann Programme, die die Kalender-Routinen benutzen, wie zum Beispiel die Berechnung des Biorhythmus. In einer Reihe von Programmmen für den privaten Bereich werden wir einige menügesteuerte Programme für diverse Berechnungen und Umrechnungen kennenlernen. Mit weiteren Programmen für technisch-wissenschaftliche und kommerzielle Anwendungen wird die Perspektive hin zum professionellen Einsatz Ihres TI-99/4A aufgezeigt.

DATENEINGABE UND MENÜTECHNIK

Im allgemeinen können Daten in Zusammenhang mit einem konkreten Problem nur Werte aus einem genau definierten Bereich annehmen. Wenn zum Beispiel die Variablen T, M und J den Tag, den Monat und das Jahr eines Kalenderdatums bezeichnen, dann kann T nur Werte aus dem Bereich von 1 bis maximal 31, M nur Werte zwischen 1 und 12 und J nur Werte entsprechend den Jahreszahlen, etwa 1900 bis 2000, anneh-

men. Dabei ist der genaue Wertebereich von T wiederum vom Monat und speziell für den Februar auch noch vom Jahr abhängig. Ein Datum 31. 6. 1983 oder 29. 2. 1983 ist sinnlos! Es ließe sich hier eine beliebige Reihe von Beispielen aufführen. Worauf es hier jedoch ankommt, ist die Tatsache, daß in guten Programmen dafür gesorgt wird, daß unsinnige Daten, soweit sie als solche zu erkennen sind, gar nicht erst weiterverarbeitet werden. In komfortableren Programmiersprachen wie etwa Pascal gibt es Hilfsmittel, mit denen die erlaubten Wertebereiche genau beschrieben werden können, und das System sorgt dann dafür, daß bei Überschreiten der erlaubten Wertegrenzen die Ausführung des Programms abgebrochen wird. In BASIC besteht diese Möglichkeit nicht. Üblicherweise gibt es nur Zahlen und Zeichen als einfache Datentypen und Felder und Strings als zusammengesetzte Typen. Es ist deshalb um so wichtiger, bei der Programmierung immer wieder Kontrollen einzubauen, die die erhaltenen Werte auf Zulässigkeit überprüfen.

Eine bewährte Methode ist es, alle eingelesenen Werte an passender Stelle auch wieder auszudrucken, damit man sofort sehen kann, mit welchen Werten eine Rechnung ausgeführt wurde. Falls es Kriterien gibt, die bereits bei der Eingabe von Daten angewendet werden können, soll dies auch sofort erfolgen. Bei der Eingabe eines nicht zulässigen Wertes soll dieser sofort reklamiert werden und die Eingabeaufforderung wiederholt werden. Mit der im folgenden beschriebenen Routine WOCHENTAG kann der zu einem durch Tag, Monat und Jahr bestimmten Datum gehörende Wochentag ermittelt werden. In dieser Routine sind Tests auf Zulässigkeit des angegebenen Datums enthalten. Im Fehlerfall wird als Wochentag der Wert 0 bzw. der String „??“ zurückgegeben. In einem Programm, in dem mit Kalenderdaten gearbeitet wird, empfiehlt es sich daher, mit jedem eingelesenen Datum sofort das Unterprogramm WOCHENTAG aufzurufen. Wird der Wert 0 als Wochentag zurückgegeben, so ist das eingelesene Datum nicht zulässig, und die Eingabe muß wiederholt werden.

Eine Grobüberprüfung kann durch Benutzung der ACCEPT-Anweisung im Extended BASIC durchgeführt werden. Mit den Optionen VALIDATE und SIZE können die erlaubten Zeichen und die Länge der Eingabe festgelegt werden. Der Versuch einer unzulässigen Eingabe wird durch einen Ton angezeigt, und das getippte Zeichen wird nicht akzeptiert. Wenn immer es möglich ist, empfiehlt sich die Verwendung dieser Eingabeanweisung.

KALENDER-UNTERPROGRAMME

In der Astronomie bezeichnet man mit „Tropisches Jahr" oder „Sonnenjahr" die Zeit, die vergeht, bis die Sonne wieder in einem der beiden Äquinoktialpunkte steht, bis also wieder die Tag-und-Nacht-Gleiche entweder im Frühling oder im Herbst erreicht ist. Dieses so definierte Jahr liegt dem Kalender und damit allen Einrichtungen des bürgerlichen Lebens zugrunde. Seine Länge beträgt genau 365 Tage 5 Stunden 48 Minuten und 45,99 Sekunden oder 365.242199 Tage. Dadurch, daß das Sonnenjahr 365 Tage und rund ¼ Tag lang ist, sind laufend Korrekturen erforderlich. Im Jahre 46 v. Chr. führte Julius Caesar einen neuen Kalender (den julianischen) ein, der nach drei Jahren mit 365 Tagen ein Schaltjahr mit 366 Tagen einfügte, um so das bürgerliche Jahr dem Sonnenverlauf anzupassen. Nun ist aber das Sonnenjahr nicht genau 365.25 Tage lang, sondern etwas kürzer, nämlich genau um 0.0078 Tage. Diese zunächst klein erscheinende Differenz läuft aber durchaus zu merkbaren Unterschieden auf. Nach 128 Jahren beträgt der Unterschied bereits einen Tag. Es mußte also eine weitere Korrektur angebracht werden. Die längst fällige Kalenderreform wurde dann 1582 von Papst Gregor XIII. durch Einführen unseres heutigen gregorianischen Kalenders durchgeführt. Durch päpstliches Dekret ließ man 10 Tage fortfallen, so daß auf Donnerstag, den 4. Oktober 1582, Freitag, der 15. Oktober 1582 folgte. Die Schaltjahre aus dem julianischen Kalender wurden beibehalten, jedoch um folgende Ausnahmen ergänzt: Alle vollen Jahrhunderte, also die Jahre 1700, 1800 usw. bleiben normale Jahre mit 365 Tagen, die Jahrhunderte aber, die durch 400 teilbar sind also 1600 und 2000, bleiben Schaltjahre. Damit wurde eine recht gute Annäherung an den Sonnenverlauf erreicht. Bis heute (1983) ist die Abweichung auf etwa 2.9 Stunden angewachsen und wird erst im Jahr 4900 einen Tag ausmachen.

Nach diesem Ausflug in die Astronomie wollen wir die Kalenderregeln in konkrete Programme umsetzen. Als erstes schreiben wir ein Unterprogramm, das uns für jeden Monat eines jeden Jahres dessen Länge angibt. Mit Ausnahme des Februars, der eine Sonderbehandlung braucht, sind bis einschließlich Juli die ungeraden Monate 31 Tage lang und die geraden 30 Tage lang. Ab August ist es genau umgekehrt. Durch Monat modulo 2, also den Rest, der bei der Division durch zwei bleibt (0 oder 1), bekommen wir bis einschließlich 7 gleich Juli einen Wert 0 oder 1, den wir zu 30 dazuaddieren, um die Monatslänge zu erhalten. Für die Monate nach dem Juli wird einfach 7 subtrahiert und

dann ebenfalls durch modulo 2 der passende Wert 0 oder 1 bestimmt. Für den Februar ist eine besondere Behandlung erforderlich, die die Jahreszahl berücksichtigt. Durch Jahreszahl modulo 4 werden die Schaltjahre ermittelt und, wenn nötig, durch die Berechnung von modulo 100 und modulo 400 die Ausnahmen bestimmt.

In den Unterprogrammen für die Kalenderrechnung sind jeweils am Anfang Überprüfungen der eingegebenen Parameter eingebaut. Es ist dies stets eine Abfrage, ob der Wert für MONAT im Bereich 1 bis 12 liegt, und eine Abfrage, ob der Wert für TAG positiv ist. Der Wert 0 wird zum Teil als Anzeige für einen Fehler benutzt. In Anpassung an den Gebrauch im täglichen Leben können Jahreszahlen im Bereich von 1900 bis 1999 auch in der zweistelligen Form 00 bis 99 angegeben werden, da der gregorianische Kalender ohnehin nur für Zeiten nach dem 15. Oktober 1582 zu korrekten Ergebnissen führt.

```
30000 REM =================
30010 REM
30020 SUB MON_LNG(MONAT,JAHR,TPM)
30030 REM
30040 REM =================
30050 REM
30060 REM
30070 DEF MODJ(Q)=J-Q*INT(J/Q)
30080 REM -------
30090 REM
30100 M=MONAT
30110 TPM=0
30120 IF M<1 OR M>12 THEN SUBEXIT
30130 J=JAHR
30140 IF J>=0 AND J<100 THEN J=J+1900
30150 IF M<>2 THEN 30220
30160 ML=28
30170 IF MODJ(4)<>0 THEN 30260
30180 IF MODJ(100)<>0 THEN 30200
30190 IF MODJ(400)<>0 THEN 30260
30200 ML=29
30210 GOTO 30260
30220 MM=M
30230 IF MM>7 THEN MM=MM-7
30240 ML=MM-2*INT(MM/2)
30250 ML=ML+30
30260 TPM=ML
30270 SUBEND
30280 REM
30290 REM
30300 REM *****************
30310 REM
```

Abb. 4.1: Unterprogramm zur Berechnung der Tage pro Monat

Das alles ist in dem Extended BASIC-Unterprogramm MON_LNG enthalten, in dem die Parameter MONAT und JAHR die Eingabeparameter sind und aus dem man das Ergebnis durch den Parameter TPM (Tage pro Monat) zurückerhält.

Bei vielen Problemen ist es einfacher, zunächst mit der Zahl der Tage zu rechnen, anstelle der üblichen Datumsangabe mit Tag, Monat, Jahr. Dafür braucht man ein Programm, das aus einer solchen Angabe den x-ten Tag des Jahres errechnet. Dafür schreiben wir uns das nachfolgende Unterprogramm TAG_IM_JAHR, das sich selbst erklärt.

```
30320 REM =================
30330 REM
30340 SUB TAG_IM_JAHR(TAG,MONAT,JAHR,TIJ)
30350 REM
30360 REM =================
30370 REM
30380 REM
30390 TIJ=0
30400 IF MONAT<1 OR MONAT>12 THEN SUBEXIT
30410 IF TAG<1 THEN SUBEXIT
30420 J=JAHR
30430 D=0
30440 FOR M=1 TO MONAT-1
30450 CALL MON_LNG(M,J,ML)
30460 D=D+ML
30470 NEXT M
30480 REM
30490 REM
30500 D=D+TAG
30510 CALL MON_LNG(MONAT,J,ML)
30520 IF TAG>ML THEN D=0
30530 TIJ=D
30540 SUBEND
30550 REM
30560 REM
30570 REM *****************
30580 REM
```

Abb. 4.2: Unterprogramm zur Berechnung des x-ten Tages im Jahr

Beim Rechnen mit Tagen braucht man des öfteren auch die Differenz zwischen zwei Daten in Tagen. Das Unterprogramm DATDIFF erhält zwei Datumsangaben und berechnet die Differenz Datum2 − Datum1 in Tagen. Das Ergebnis wird über den Parameter DIFF zurückgegeben. Der Parameter FEHLER wird auf Null gesetzt, wenn keine Fehler festgestellt wurden, im Fehlerfall enthält er den Wert 1 oder 2. Es wird

zunächst einmal der jeweilige Tag im Jahr bestimmt. Wenn beide Daten im gleichen Jahr liegen, kann die Differenz unmittelbar ausgerechnet werden. Liegen die Daten in verschiedenen Jahren, dann muß die Anzahl der Tage der dazwischen liegenden Jahre berechnet werden, und es müssen dann noch die restlichen Tage der beiden betroffenen Jahre dazuaddiert werden.

Bei der Berechnung der Tage der dazwischen liegenden Jahre muß berücksichtigt werden, in welcher Reihenfolge die Daten liegen. Dementsprechend ist die Differenz als positiver oder negativer Wert auszugeben.

```
30590 REM =================
30600 REM
30610 SUB DATDIFF(T1,M1,JAHR1,T2,M2,JAHR2,DIFF,FEHLER)
30620 REM
30630 REM =================
30640 REM
30650 REM
30660 J1=JAHR1 :: J2=JAHR2
30670 IF J1>=0 AND J1<100 THEN J1=J1+1900
30680 IF J2>=0 AND J2<100 THEN J2=J2+1900
30690 CALL TAG_IM_JAHR(T1,M1,J1,DOY1)
30700 CALL TAG_IM_JAHR(T2,M2,J2,DOY2)
30710 DIFF=-1
30720 FEHLER=0
30730 IF DOY1<=0 THEN FEHLER=1
30740 IF DOY2<=0 THEN FEHLER=2
30750 IF FEHLER<>0 THEN SUBEXIT
30760 REM
30770 REM -----------------
30780 REM
30790 DIFF=0
30800 IF J1=J2 THEN DIFF=DOY2-DOY1 :: SUBEXIT
30810 REM
30820 REM -----------------
30830 REM
30840 IF J1>J2 THEN 30920
30850 SJ=J1+1
30860 EJ=J2-1
30870 SIG=1
30880 GOTO 30990
30890 REM
30900 REM -----------------
30910 REM
30920 SJ=J2+1
30930 EJ=J1-1
30940 SIG=-1
30950 D=DOY1 :: DOY1=DOY2 :: DOY2=D
30960 REM
30970 REM -----------------
30980 REM
30990 DIFF=0
```

```
31000 IF SJ>EJ THEN 31090
31010 DIFF=(EJ-SJ+1)*365
31020 FOR J=SJ TO EJ
31030 CALL MON_LNG(2,J,ML)
31040 IF ML=29 THEN DIFF=DIFF+1
31050 NEXT J
31060 REM
31070 REM ----------------
31080 REM
31090 DIFF=DIFF+DOY2
31100 DIFF=DIFF+365-DOY1
31110 CALL MON_LNG(2,SJ-1,ML)
31120 IF ML=29 THEN DIFF=DIFF+1
31130 DIFF=DIFF*SIG
31140 SUBEND
31150 REM
31160 REM
31170 REM ****************
31180 REM
```

Abb. 4.3: Unterprogramm zur Berechnung der Differenz zwischen zwei Daten

Als nächstes schreiben wir uns für Kalenderarbeiten ein Unterprogramm, mit dem zu jedem Datum der Wochentag bestimmt wird. Dazu gehen wir folgendermaßen vor: Ausgehend von einem bekannten Datum, wir wählen dafür den 15. Oktober 1983, der auf einen Samstag fiel, wird die Differenz in Tagen zu dem zu bearbeitenden Datum ausgerechnet. Diese Differenz modulo 7 liefert uns dann die Verschiebung gegenüber dem Samstag. Die so erhaltene Verschiebung muß nun noch so transformiert werden, daß man die Angabe des Wochentages in der genormten Form erhält, mit 1 für Montag bis 7 für Sonntag. Dabei ist zu beachten, daß die Differenz, die wir aus der Routine DATDIFF erhalten, positiv oder negativ sein kann, je nachdem, ob das zu untersuchende Datum nach oder vor dem 15. 10. 1983 liegt. DIFF modulo 7 nimmt damit die Werte von − bis + 6 an. Die folgende Tabelle macht die Transformation deutlich:

1	SO	MO	DI	MI	DO	FR	SA	SO	MO	DI	MI	DO	FR
2	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6
3	6	7	8	9	10	11	12	13	14	15	16	17	18
4	6	0	1	2	3	4	5	6	0	1	2	3	4
5	7	1	2	3	4	5	6	7	1	2	3	4	5

Abb. 4.4: Wochentag-Tabelle

Ausgehend von Samstag, dem 15. Oktober 1983, ergeben sich für DIFF modulo 7 die in Zeile 2 aufgeführten Werte in dieser Zuordnung zu den Wochentagen in der Zeile 1. Die Transformation muß so festgelegt werden, daß alle Werte positiv werden, und daß für den ersten Montag in Zeile 1 ein Wert herauskommt, der modulo 7 gleich Null ist. Durch Addition von 5 könnte die letzte Bedingung erfüllt werden, jedoch bleibt dann der erste Wert (für Sonntag) negativ, nämlich −1. Es muß daher mit einer um 7 größeren Zahl addiert werden. Das Ergebnis der Addition mit 12 zeigt die Zeile 3. Rechnet man diese Werte modulo 7, so ergibt das die Zeile 4, und dazu 1 addiert, erhält man schließlich die gewünschten Werte in der Zeile 5.

```
31190 REM =================
31200 REM
31210 SUB WOCHENTAG(T,M,J,W,W$)
31220 REM
31230 REM =================
31240 REM
31250 REM
31260 DEF MOD7(X)=X-7*INT(X/7)
31270 REM -------
31280 REM
31290 CALL DATDIFF(15,10,83,T,M,J,DIFF,FEHLER)
31300 W=0 :: W$="??"
31310 IF FEHLER<>0 THEN SUBEXIT
31320 DIFF=MOD7(DIFF)+12
31330 W=MOD7(DIFF)+1
31340 ON W GOTO 31350,31360,31370,31380,31390,31400,31410
31350 W$="MO" :: SUBEXIT
31360 W$="DI" :: SUBEXIT
31370 W$="MI" :: SUBEXIT
31380 W$="DO" :: SUBEXIT
31390 W$="FR" :: SUBEXIT
31400 W$="SA" :: SUBEXIT
31410 W$="SO"
31420 SUBEND
31430 REM
31440 REM
31450 REM *****************
31460 REM
```

Abb. 4.5: Unterprogramm zur Bestimmung des Wochentages

Bei terminbezogenen Problemen ergibt sich des öfteren die Aufgabe, zu einem Datum in der Form Tag, Monat, Jahr eine Anzahl von Tagen zu addieren oder davon zu subtrahieren. Das Ergebnis wird dann wieder in der Form Tag, Monat, Jahr gebraucht. Die nachfolgende Subroutine

TERMIN hat als Eingabeparameter T1, M1, J1 und TX und liefert als Ergebnis das Datum in der Form T2, M2, J2. Dabei kann die Anzahl der Tage TX sowohl positiv als auch negativ sein.

```
31470 REM =================
31480 REM
31490 SUB TERMIN(T1,M1,J1,TX,T2,M2,J2)
31500 REM
31510 REM =================
31520 REM
31530 REM
31540 IF TX<0 THEN 32090
31550 REM
31560 REM -----------------
31570 REM    TX ADDIEREN
31580 REM -----------------
31590 REM
31600 IF TX=0 THEN T2=T1 :: M2=M1 :: J2=J1 :: SUBEXIT
31610 CALL TAG_IM_JAHR(T1,M1,J1,TIJ1)
31620 NTIJ=365-TIJ1
31630 CALL MON_LNG(M1,J1,ML)
31640 IF ML=29 THEN NTIJ=NTIJ+1
31650 IF TX>NTIJ THEN 31860
31660 REM
31670 REM -----------------
31680 REM TERMIN LIEGT IM
31690 REM GLEICHEN JAHR
31700 REM -----------------
31710 REM
31720 J2=J1
31730 IF TX=NTIJ THEN T2=31 :: M2=12 :: SUBEXIT
31740 CALL MON_LNG(M1,J1,ML)
31750 NTIM=ML-T1
31760 IF TX<=NTIM THEN T2=T1+TX :: M2=M1 :: SUBEXIT
31770 TR=TX
31780 M2=M1
31790 REM
31800 TR=TR-NTIM :: M2=M2+1
31810 CALL MON_LNG(M2,J2,NTIM)
31820 IF TR>NTIM THEN 31790
31830 T2=TR
31840 SUBEXIT
31850 REM
31860 REM -----------------
31870 REM   TERMIN LIEGT IN
31880 REM   EINEM SPAETEREN
31890 REM   JAHR
31900 REM -----------------
31910 REM
31920 TR=TX :: J2=J1 :: M2=1 :: T2=1
31930 REM
31940 TR=TR-NTIJ :: J2=J2+1
31950 NTIJ=365
31960 CALL MON_LNG(2,J2,ML)
31970 IF ML=29 THEN NTIJ=366
31980 IF TR>NTIJ THEN 31930
```

```
31990 IF TR=NTIJ THEN T2=31 :: M2=12 :: SUBEXIT
32000 IF TR<31 THEN T2=TR :: M2=1 :: SUBEXIT
32010 NTIM=31
32020 GOTO 31790
32030 REM
32040 REM
32050 REM -----------------
32060 REM  TX SUBTRAHIEREN
32070 REM -----------------
32080 REM
32090 TX=ABS(TX)
32100 CALL TAG_IM_JAHR(T1,M1,J1,TIJ1)
32110 IF TX>=TIJ1 THEN 32290
32120 REM
32130 REM -----------------
32140 REM   TERMIN LIEGT IM
32150 REM   GLEICHEN JAHR
32160 REM -----------------
32170 REM
32180 J2=J1
32190 IF TX<T1 THEN T2=T1-TX :: M2=M1 :: SUBEXIT
32200 NTIM=T1
32210 TR=TX
32220 M2=M1
32230 REM
32240 TR=TR-NTIM :: M2=M2-1
32250 CALL MON_LNG(M2,J2,NTIM)
32260 IF TR>NTIM THEN 32230
32270 T2=NTIM-TR
32280 SUBEXIT
32290 REM
32300 REM -----------------
32310 REM   TERMIN LIEGT IN
32320 REM   EINEM FRUEHEREN
32330 REM   JAHR
32340 REM -----------------
32350 REM
32360 TR=TX :: J2=J1 :: M2=12 :: T2=31 :: NTIJ=TIJ1
32370 REM
32380 TR=TR-NTIJ :: J2=J2-1
32390 NTIJ=365
32400 CALL MON_LNG(2,J2,ML)
32410 IF ML=29 THEN NTIJ=366
32420 IF TR>NTIJ THEN 32370
32430 IF TR=NTIJ THEN T2=1 :: M2=1 :: SUBEXIT
32440 NTIM=31
32450 M2=12
32460 GOTO 32260
32470 SUBEND
32480 REM
32490 REM
32500 REM *****************
32510 REM
```

Abb. 4.6: Unterprogramm zur Berechnung eines terminbezogenen Datums

Das nachfolgende kleine Testprogramm demonstriert die Benutzung der Kalender-Unterprogramme. Es empfiehlt sich, die Kalenderroutinen, wie auf den vorigen Seiten angegeben, alle mit Zeilennummern ab 30000 zu schreiben und dann auf einen externen Speicher zu sichern. Wenn Sie diese Unterprogrammsammlung auf Magnetbandkassette abgespeichert haben, können Sie diese zuerst vom Band laden und davor das Hauptprogramm TERMINRECHNUNG schreiben.

```
100 REM =================
110 REM
120 REM  TERMINRECHNUNG
130 REM
140 REM =================
150 REM
160 REM
170 PRINT :: PRINT
180 PRINT "===========================" :: PRINT :: PRINT
190 REM
200 REM
210 INPUT "DATUM = ":T1,M1,J1
220 IF T1=0 THEN STOP
230 INPUT "TAGE = ":TX
240 CALL WOCHENTAG(T1,M1,J1,W1,W1$)
250 IF W1<>0 THEN 290
260 PRINT "DAS DATUM ";T1;M1;J1;" GIBT ES NICHT"
270 PRINT " "
280 GOTO 210
290 REM
300 CALL TERMIN(T1,M1,J1,TX,T2,M2,J2)
310 PRINT :: PRINT
320 CALL WOCHENTAG(T2,M2,J2,W2,W2$)
330 PRINT T2;M2;J2;W2$
340 GOTO 160
350 REM
360 REM
370 REM *****************
380 REM
```

Abb. 4.7: Hauptprogramm Terminrechnung

BIORHYTHMUS

Als eine Anwendung der Kalender-Unterprogramme zeigen wir Ihnen jetzt ein Programm zur Berechnung der Kurven des Biorhythmus.
Aus der täglichen Erfahrung ist jedem bekannt, daß es in allen Bereichen des Lebens Auf- und Ab-Bewegungen gibt, die eine gewisse Regelmäßigkeit besitzen wie der Tagesablauf oder die Abfolge der

Jahreszeiten. Um die Jahrhundertwende hat sich der Berliner Biologe und Arzt Dr. W. Fliess näher damit beschäftigt und derartige Rhythmen untersucht. Auf Grund seiner Ergebnisse und der Arbeiten, die daraufhin auch von anderen unternommen wurden, geht man heute von drei Zyklen aus:

- dem physischen Zyklus mit einer Periode von 23 Tagen, der sich auf Energie, Angriffslust, Unternehmungsgeist, Selbstvertrauen, Mut, physische Kraft, Ausdauer und Widerstandskraft bezieht;
- dem sensitiven Zyklus mit einer Periode von 28 Tagen, der sich auf Gefühl, Stimmung, Optimismus, Frohsinn, Gemeinschaftssinn, Kunst, Intuition und schöpferische Fähigkeiten bezieht;
- dem intellektuellen Zyklus mit einer Periode von 33 Tagen, der sich auf Verstandesklarheit, Auffassungskraft, Wiedergabefähigkeit, Geistesgegenwart, Schlagfertigkeit in Rede und Schrift, Kombinationsgabe und Logik bezieht.

Nach der Auffassung der Vertreter des Biorhythmus beginnen alle drei Zyklen im Moment der Geburt und schwingen von da an das ganze Leben hindurch jeweils mit ihrer stets gleichen Periode. Jeder dieser Zyklen besteht aus einem Plus- oder Hochtriebbereich und einem Minus- oder Niedertriebbereich. In der Minusphase sinkt die jeweilige Leistungsfähigkeit ab, man hat eine Auffüll- oder Erholungsphase, während man in der Plusphase auch höchsten Beanspruchungen besser gewachsen sein soll. Die Tage mit den Nulldurchgängen gelten als kritische Tage, in denen eine besondere Neigung zu körperlichem und geistigem Versagen bestehen soll.

Diese Bemerkungen zum Biorhythmus sollen nur dazu dienen, Ihnen das nachfolgende Programm verständlich zu machen. Sie stellen keineswegs eine Stellungnahme zu dieser Theorie dar. Der daran interessierte Leser wird auf die einschlägige Literatur verwiesen. Wir wollen uns hier lediglich mit der technischen Seite der Erstellung und Darstellung dieser Kurven befassen. Eine mathematisch geeignete Form der Beschreibung eines solchen Verhaltens stellen die Sinusfunktionen mit der entsprechenden Periode dar, die alle drei mit dem Tag der Geburt bei Null beginnen. Ein Programm für die Berechnung des Biorhythmus muß also, beginnend vom Geburtstag, die Anzahl der Tage bis zu dem gewünschten Datum bestimmen und daraus den Verlauf der drei Sinuskurven berechnen und darstellen.

Einige Erläuterungen zum Programm:

Die am Anfang definierten Faktoren FP, FS und FI dienen zur Berechnung der Sinuswerte. Sie bilden die Tage auf die erforderlichen Perioden von 23, 28 und 33 Tagen ab, so daß diese Perioden jeweils 2 * pi, der Periode des Sinus, entsprechen. In den folgenden Anweisungen (Zeilen 1160 bis 1240) werden die Patterns der Buchstaben P, S und I ermittelt und damit neue Zeichen mit den Codenummern 127, 130 und 140 erzeugt. Dies geschieht, damit die drei Zeichen in verschiedenen Zeichengruppen sind und so in unterschiedlichen Farben dargestellt werden können. Die Anweisungen 1300 bis 1680 produzieren das Startbild auf den Bildschirm und fragen die Daten für den laufenden Tag, den Geburtstag und den ersten Tag ab, für den die Kurven berechnet werden sollen. Es werden, bedingt durch die Spaltenzahl auf dem Bildschirm, immer die Kurven für vier Wochen berechnet. Mittels der ACCEPT-Anweisung und der Parameter VALIDATE und SIZE werden die Eingaben kontrolliert. Durch Aufruf der Routine WOCHENTAG wird darüber hinaus das Datum auf Zulässigkeit geprüft.
Nach dieser Eingabeprozedur werden, sozusagen als Nebenprodukt, Ihr auf den laufenden Tag bezogenes Alter in Tagen und der Wochentag Ihres Geburtstages angezeigt. Nach Drücken eine beliebigen Taste (Zeile 1890) wird dann der Kurvenverlauf berechnet und dargestellt. Dazu wird in den Zeilen 1970 bis 2050 ein Raster mit einer Wocheneinteilung auf den Bildschirm geschrieben, in das dann die Sinuskurven eingezeichnet werden. Die Umrechnung der Tagesangaben und der Y-Werte in Zeilen- und Spaltennummern erfolgt in den Parametern der HCHAR-Routine in den Zeilen 2190 bis 2210. Schließlich werden danach noch die Wochentage und die Tages- und Monatsangabe an die entsprechenden Stellen geschrieben. Auf Wunsch kann das Verfahren mehrmals wiederholt werden.

```
1000 REM =================
1010 REM
1020 REM        BIORHYTHMUS
1030 REM
1040 REM
1050 REM =================
1060 REM
1070 REM
1080 REM
```

```
1090 REM -----------------
1100 REM INITIALISIERUNGEN
1110 REM -----------------
1120 REM
1130 FP=(2*PI)/23
1140 FS=(2*PI)/28
1150 FI=(2*PI)/33
1160 CALL CHARPAT(80,P$)
1170 CALL CHARPAT(83,S$)
1180 CALL CHARPAT(73,I$)
1190 CALL CHAR(127,P$)
1200 CALL CHAR(130,S$)
1210 CALL CHAR(140,I$)
1220 CALL COLOR(12,3,1)
1230 CALL COLOR(13,8,1)
1240 CALL COLOR(14,14,1)
1250 REM
1260 REM -----------------
1270 REM       EINLESEN
1280 REM -----------------
1290 REM
1300 CALL CLEAR
1310 DISPLAY AT(5,5):"B I O R H Y T H M U S"
1320 DISPLAY AT(7,5):"=*=*=*=*=*=*=*=*=*=*="
1330 DISPLAY AT(11,3):"DATUMSANGABEN IN DER FORM"
1340 DISPLAY AT(12,3):"TT MM JJ  oder TT MM JJJJ"
1350 REM
1360 REM -----------------
1370 REM        HEUTE
1380 REM -----------------
1390 REM
1400 DISPLAY AT(14,1):"HEUTE IST DER "
1410 ACCEPT AT(14,16)VALIDATE(DIGIT)SIZE(2):TH
1420 ACCEPT AT(14,19)VALIDATE(DIGIT)SIZE(2):MH
1430 ACCEPT AT(14,22)VALIDATE(DIGIT)SIZE(4):JH
1440 CALL WOCHENTAG(TH,MH,JH,WH,WH$)
1450 IF WH<>0 THEN 1540
1460 DISPLAY AT(14,1):"DIESES DATUM GIBT ES NICHT"
1470 CALL SOUND(750,30000,30):: CALL SOUND(1,30000,30)
1480 GOTO 1400
1490 REM
1500 REM -----------------
1510 REM      GEBURTSTAG
1520 REM -----------------
1530 REM
1540 DISPLAY AT(18,1):"GEBURTSTAG  : "
1550 ACCEPT AT(18,16)VALIDATE(DIGIT)SIZE(2):TG
1560 ACCEPT AT(18,19)VALIDATE(DIGIT)SIZE(2):MG
1570 ACCEPT AT(18,22)VALIDATE(DIGIT)SIZE(4):JG
1580 CALL WOCHENTAG(TG,MG,JG,WG,WG$)
1590 IF WG<>0 THEN 1680
1600 DISPLAY AT(18,1):"DIESES DATUM GIBT ES NICHT"
1610 CALL SOUND(750,30000,30):: CALL SOUND(1,30000,30)
1620 GOTO 1540
1630 REM
1640 REM -----------------
1650 REM     START DATUM
1660 REM -----------------
1670 REM
```

```
1680 DISPLAY AT(20,1):"START DATUM :  "
1690 ACCEPT AT(20,16)VALIDATE(DIGIT)SIZE(2):TS
1700 ACCEPT AT(20,19)VALIDATE(DIGIT)SIZE(2):MS
1710 ACCEPT AT(20,22)VALIDATE(DIGIT)SIZE(4):JS
1720 CALL WOCHENTAG(TS,MS,JS,WS,WS$)
1730 IF WS<>0 THEN 1810
1740 DISPLAY AT(20,1):"DIESES DATUM GIBT ES NICHT"
1750 CALL SOUND(750,30000,30):: CALL SOUND(1,30000,30)
1760 GOTO 1680
1770 REM
1780 REM -----------------
1790 REM   ALTER IN TAGEN
1800 REM -----------------
1810 REM
1820 CALL DATDIFF(TG,MG,JG,TH,MH,JH,ALTER,F)
1830 CALL HCHAR(11,1,32,450)
1840 DISPLAY AT(12,1):"SIE SIND HEUTE ";ALTER
1850 DISPLAY AT(13,1):"TAGE ALT. IHR GEBURTSTAG"
1860 DISPLAY AT(14,1):"WAR EIN ";WG$
1870 DISPLAY AT(23,1):"ZUM  FORTSETZEN EINE "
1880 DISPLAY AT(24,1):"BELIEBIGE TASTE DRUECKEN"
1890 CALL KEY(0,Z,S)
1900 IF S=0 THEN 1890
1910 REM
1920 REM -----------------
1930 REM RASTER MIT WOCHEN-
1940 REM EINTEILUNG
1950 REM -----------------
1960 REM
1970 CALL CLEAR
1980 CALL HCHAR(4,3,45,28)
1990 CALL HCHAR(11,3,45,28)
2000 CALL HCHAR(18,3,45,28)
2010 FOR DD=0 TO 28 STEP 7
2020 CALL HCHAR(4,DD+3,43,1)
2030 CALL HCHAR(11,DD+3,43,1)
2040 CALL HCHAR(18,DD+3,43,1)
2050 NEXT DD
2060 REM
2070 REM -----------------
2080 REM KURVEN RECHNEN
2090 REM UND ZEICHNEN
2100 REM -----------------
2110 REM
2120 CALL DATDIFF(TG,MG,JG,TS,MS,JS,DIFF,F)
2130 DIFF=ABS(DIFF)
2140 FOR DD=0 TO 28
2150 D=DIFF+DD
2160 YP=SIN(D*FP)
2170 YS=SIN(D*FS)
2180 YI=SIN(D*FI)
2190 CALL HCHAR(11-YP*6,DD+3,80,1)
2200 CALL HCHAR(11-YS*6,DD+3,83,1)
2210 CALL HCHAR(11-YI*6,DD+3,73,1)
2220 NEXT DD
2230 REM
2240 REM -----------------
2250 REM BESCHRIFTEN
2260 REM -----------------
```

```
2270 REM
2280 FOR DD=0 TO 21 STEP 7
2290 CALL TERMIN(TS,MS,JS,DD,T,M,J)
2300 CALL WOCHENTAG(T,M,J,W,W$)
2310 CALL HCHAR(19,DD+3,124,1)
2320 DISPLAY AT(20,DD+1):USING "":W$
2330 DISPLAY AT(21,DD+1):USING 2340:T,M
2340 IMAGE
2350 DISPLAY AT(21,DD+3):".";
2360 NEXT DD
2370 REM
2380 REM -----------------
2390 REM FERTIG. NOCH MAL?
2400 REM -----------------
2410 REM
2420 DISPLAY AT(24,1):"NOCH EINMAL? (J/N)"
2430 ACCEPT AT(24,20)VALIDATE("JN"):JN$
2440 IF JN$="J" THEN 1300
2450 STOP
2460 REM
2470 REM
2480 REM *****************
2490 REM
```

Abb. 4.8: Programm-Listing Biorhythmus

BALKENDIAGRAMME

In vielen Fällen sind Listen oder Tabellen mit Zahlen nicht so gut geeignet, das, was sie beschreiben, auch entsprechend deutlich erkennen zu lassen. In solchen Fällen ist eine grafisch aufbereitete Information wesentlich besser zu erfassen. Wir zeigen Ihnen im folgenden zwei Verfahren dafür. Das erste ist die Darstellung als Balkendiagramm, auch Säulendiagramm oder Histogramm genannt. Dabei werden die Werte durch die Höhe von nebeneinanderstehenden Balken repräsentiert und gestatten so, auf einen Blick die Verhältnisse zu erfassen.
Bedingt durch die Bildschirmgröße, können mit dem folgenden Programm bis zu 12 Balken dargestellt werden, deren maximale Höhe 20 Zeilen sein kann. Um auch andere Wertebereiche darstellen zu können, werden die eingegebenen Werte so normiert, daß der maximale Wert gerade 20 ergibt. Alle anderen werden in dem so bestimmten Verhältnis dargestellt.
Im Teil Initialisierungen werden mittels der CHAR-Routine Zeichen für den Aufbau der Balken definiert und mit COLOR ihre Farben festgelegt. Im Teil Einlesen des Programms kann ausgewählt werden, wie viele Balken gezeichnet werden sollen, und anschließend werden

die erforderlichen Werte der Reihe nach eingelesen. Für die nachfolgende Normierung wird zunächst der maximale Wert gesucht und damit der Faktor AMAXR bestimmt, mit dem dann alle Werte multipliziert werden. Zum Zeichnen der Balken wird das Unterprogramm VCHAR benutzt, bei dem als letzter Parameter ein Wiederholungsfaktor angegeben werden kann. Dieser Faktor ist gerade der jeweils darzustellende normierte Wert.
Damit das Bild erhalten bleibt, wird am Ende des Programms mit einer Endlosschleife der Rechner so lange beschäftigt, bis eine beliebige Taste gedrückt wird.

```
1000 REM =================
1010 REM
1020 REM  BALKENDIAGRAMME
1030 REM      ZEICHNEN
1040 REM
1050 REM
1060 REM =================
1070 REM
1080 REM
1090 REM
1100 REM
1110 REM -----------------
1120 REM INITIALISIERUNGEN
1130 REM -----------------
1140 REM
1150 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
1160 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
1170 CALL CHAR(105,"0103070F1F3F7FFF")
1180 CALL CHAR(112,"0103070F1F3F7FFF")
1190 CALL CHAR(121,"0103070F1F3F7FFF")
1200 CALL CHAR(130,"0103070F1F3F7FFF")
1210 CALL COLOR(9,4,1)
1220 CALL COLOR(10,5,8)
1230 CALL COLOR(11,1,5)
1240 CALL COLOR(12,8,1)
1250 CALL COLOR(13,1,8)
1260 CALL SCREEN(13)
1270 REM
1280 REM
1290 REM -----------------
1300 REM      EINLESEN
1310 REM -----------------
1320 REM
1330 REM
1340 REM
1350 HL$="----------------------------"
1360 CALL CLEAR
1370 DISPLAY AT(1,1):HL$
1380 DISPLAY AT(2,3):"BALKENDIAGRAMM ZEICHNEN"
1390 DISPLAY AT(3,1):HL$
1400 REM
```

```
1410 REM
1420 FEHLER=0
1430 CALL HCHAR(9,1,32,500)
1440 DISPLAY AT(9,1):"ES KOENNEN MAXIMAL 12 BALKEN"
1450 DISPLAY AT(10,1):"GEZEICHNET WERDEN"
1460 IF FEHLER=0 THEN 1500
1470 CALL SOUND(500,200,2)
1480 CALL SOUND(500,30000,30)
1490 CALL SOUND(1,30000,30)
1500 DISPLAY AT(13,1):"ANZAHL DER SAEULEN = ";
1510 ACCEPT AT(13,22)VALIDATE(DIGIT):ANZAHL
1520 IF ANZAHL<1 OR ANZAHL>12 THEN FEHLER=1 :: GOTO 1430
1530 REM
1540 REM ----------------
1550 REM
1560 ZEIGER=0
1570 DIM A(12)
1580 CALL HCHAR(4,1,32,500)
1590 FOR I=1 TO ANZAHL
1600 DISPLAY AT(I+5,1):USING "Y() = ":I
1610 ACCEPT AT(I+5,9):A(I)
1620 NEXT I
1630 REM
1640 REM
1650 REM ----------------
1660 REM    NORMIERUNG
1670 REM ----------------
1680 REM
1690 REM
1700 AMAX=A(1)
1710 FOR I=1 TO ANZAHL
1720 IF A(I)>AMAX THEN AMAX=A(I)
1730 NEXT I
1740 REM
1750 REM
1760 AMAXR=20/AMAX
1770 FOR I=1 TO ANZAHL
1780 A(I)=INT(A(I)*AMAXR)
1790 NEXT I
1800 REM
1810 REM
1820 REM ----------------
1830 REM     ZEICHNEN
1840 REM ----------------
1850 REM
1860 REM
1870 FOR I=1 TO ANZAHL
1880 SPALTE=I*2+4
1890 IF A(I)<1 THEN 2000
1900 HOEHE=25-A(I)
1910 SPITZE=HOEHE-1
1920 CALL VCHAR(HOEHE,SPALTE,96,A(I))
1930 CALL HCHAR(SPITZE,SPALTE,121,1)
1940 CALL VCHAR(HOEHE,SPALTE+1,104,A(I)-1)
1950 CALL HCHAR(SPITZE,SPALTE+1,105,1)
1960 CALL HCHAR(24,SPALTE+1,112,1)
1970 GOTO 2020
1980 REM
1990 REM
```

```
2000 CALL HCHAR(24,SPALTE,121,1)
2010 CALL HCHAR(24,SPALTE+1,130,1)
2020 NEXT I
2030 REM
2040 REM -----------------
2050 REM
2060 CALL KEY(0,K,S)
2070 IF S=0 THEN 2060
2080 REM
2090 REM
2100 REM *****************
2110 REM
```

Abb. 4.9: Programm zum Erstellen von Balkendiagrammen

PLOTPROGRAMM FÜR HOCHAUFLÖSENDE GRAFIK

Dieses Programm gestattet die Darstellung von Punkten auf einer Fläche. Die (x, y)-Paare können entweder durch Berechnungen bestimmt oder auch einzeln eingelesen werden. In der abgedruckten Form des Programms werden die x-Werte, beginnend mit einem Anfangswert XANFANG, mit einer Schrittweite XDELTA bis zum Endwert XENDE (vgl. den Teil Initiierungen ab Zeile 1460) durch eine FOR-Schleife nacheinander bestimmt und zu jedem x mittels der durch DEF festgelegten Funktion FKT(X) (Zeile 1130) ein y-Wert berechnet. In unserem Beispiel ist das die Sinusfunktion. Durch Ändern der Funktion FKT oder auch durch Abänderung der Bestimmung der (x, y)-Wertepaare kann das Programm ziemlich leicht der Darstellung anderer Daten angepaßt werden.

In dem Teil Normierungen des Programms wird die Anpassung der von außen vorgegebenen x- und y-Werte auf die Größenverhältnisse des Bildschirms vorgenommen. Die zur Verfügung stehende Fläche ist in der x-Richtung 25 Zeichen lang und in der y-Richtung 20 Zeichen. Die Extremwerte XA und XE der x-Werte und entsprechend die Extremwerte YA und YE der y-Werte müssen so auf die Fläche abgebildet werden, daß sie diese gerade ausfüllen. Wenn man nur diese Darstellung benutzt, also das Bild aus Kästchen aufbaut, die die Größe eines ganzen Zeichens haben, hat man auch nur ein dementsprechend kleines Auflösungsvermögen und damit ein ziemlich grob gerastertes Bild. Es gibt jedoch eine Möglichkeit, eine um den Faktor 8 höhere Auflösung zu erreichen. Im Kapitel Spielen hatten wir in einem Abschnitt die

Gestaltung eigener Zeichen ausprobiert. Die Gestalt eines Zeichens wird in einem 8×8 Raster festgelegt. Diese jeweils 64 Punkte sind die kleinsten Bildelemente (auch Pixels genannt), die man ansprechen kann. Um also eine hochauflösende Grafik zu bekommen, brauchen wir eine Methode, die uns aus den (x, y)-Wertepaaren an der richtigen Stelle des Bildschirms ein Zeichen ausgibt, das wiederum so bestimmt werden muß, daß in dem Zeichen nur die Pixels gesetzt sind, die den darzustellenden Werten entsprechen. Der erste Punkt ist ziemlich leicht zu erfüllen: Wir haben das Verfahren schon bei der Darstellung der Kurven des Biorhythmus kennengelernt. Die Koordinaten für das Zeichen sind im Programm mit IXCH und IYCH bezeichnet. Für den zweiten Teil der obigen Anforderung brauchen wir noch ein geeignetes Verfahren, das letztlich einen String von 16 Hexadezimalziffern liefert, mit dem dann das CHAR-Unterprogramm aufgerufen werden kann.

Analog zu dem ersten Teil kann innerhalb des 8×8 Rasters die Position (IX, IY) für das Bildelement bestimmt werden. Aus diesen beiden Angaben kann der String für die CHAR-Routine ermittelt werden. Das so bestimmte Zeichen würde dann jedoch nur aus genau einem Punkt bestehen. Wenn sich in der Nähe weitere Punkte befinden, die ebenfalls in diesem Raster zu liegen kommen, tritt das Problem auf, eine im Prinzip nicht vorhersehbare Anzahl von Pixels an nicht vorhersehbaren Stellen im Raster ansprechen zu müssen. Wir haben das Problem so gelöst, daß zunächst einmal die Informationen über den Aufbau des Bildes gespeichert werden, so daß zu jeder Zeit während der Verarbeitung auf die bereits vorhandenen Bildelemente zurückgegriffen werden kann. Die zweifach dimensionierte Stringvariable L$ (25, 20) kann für jede Zeichenposition auf dem Bildschirm einen String aufnehmen, der das dort darzustellende Zeichen beschreibt. Wenn an der betreffenden Stelle IXCH, IYCH etwas zu zeichnen ist, wird geprüft, ob dort bereits ein Zeichen vorhanden ist. Wenn nicht, dann wird diese Position damit erstmals besetzt. Wenn die Position bereits mit einem zuvor festgelegten Zeichen besetzt ist, muß dieses so verändert werden, daß der neue Bildpunkt miteingefügt wird. Dazu greifen wir auf das im Kapitel Lernen über Bits und Bytes Besprochene zurück. Nehmen wir uns einmal eine Vierergruppe aus dem Raster heraus. Durch eine Folge von 0 oder 1 wird angegeben, ob die betreffende Position besetzt ist. Die Folge 0010 heißt also, daß nur die dritte Stelle besetzt ist. Soll zusätzlich die erste Stelle besetzt werden, muß die Folge in 1010 umgeändert werden. Diese erhält man, indem man die erste Folge 0010 mit der Folge 1000, die ein Element nur an der ersten Stelle beschreibt, durch

ein logisches ODER miteinander verknüpft: 0010 OR 1000 ergibt 1010. Die Funktion OR in BASIC führt bitweise die Oderverknüpfung durch. Wir haben damit das gesuchte Verfahren, zu einem vorhandenen Bildpunkt noch einen weiteren hinzuzufügen. Im Programm besorgt das die Subroutine ODERN, die den vorhandenen String von Hexadezimalziffern so ändert, daß an der Stelle (IX, IY) ein zusätzlicher Bildpunkt erscheint. Die Hilfsroutinen CHTOHEX (ab Zeile 2060) und HEXTOCH (ab Zeile 2180) besorgen die Umwandlung der für das Schreiben von Hexadezimalzahlen benutzten Zeichen 0 bis 9 und A bis F in die entsprechenden Zahlenwerte und umgekehrt.

Ist auf der Position (IXCH, IYCH) noch kein Zeichen vorhanden, so wird ein Hexadezimal-Ziffernstring aufgebaut, der genau den einen Bildpunkt an der Stelle (IX, IY) erzeugt. Im Programm erfolgt das in den Zeilen 1790 und 1800. Wenn alle Wertepaare (x, y) bearbeitet wurden und damit alle Zeichen für die Darstellung des Bildes bekannt sind, wird die Ausgabe begonnen. Die Anzahl der mittels des CHAR-Unterprogramms definierbaren Zeichen begrenzt auch die Möglichkeiten des Umfanges der darstellbaren Plots. Die Variable PLCH zählt die insgesamt erzeugten Zeichen mit. Maximal können 111 Zeichen mittels CHAR neu definiert werden, nämlich die für die Codes von 32 bis 143. Im Programm wird dies durch die Variable CHNUM gesteuert, die von 143 abwärts die Codes vergibt.

```
1000 REM =================
1010 REM
1020 REM   PLOT-PROGRAMM
1030 REM
1040 REM        FUER
1050 REM  HOCHAUFLOESENDE
1060 REM       GRAPHIK
1070 REM
1080 REM =================
1090 REM
1100 REM
1110 REM
1120 REM
1130 DEF FKT(X)=SIN(F*X)
1140 REM ---------------
1150 REM
1160 REM
1170 REM
1180 DIM L$(25,20)
1190 CALL CLEAR
1200 DISPLAY AT(4,9):"PLOT PROGRAMM"
1210 REM
```

```
1220 REM -----------------
1230 REM NORMIERUNGEN
1240 REM -----------------
1250 REM
1260 XA=0
1270 XE=10
1280 YA=-1.5
1290 YE=1.5
1300 DX=(XE-XA)/200
1310 DY=(YE-YA)/160
1320 DXCH=(XE-XA)/25
1330 DYCH=(YE-YA)/20
1340 DXR=1/DX :: DXCHR=1/DXCH
1350 DYR=1/DY :: DYCHR=1/DYCH
1360 REM
1370 REM
1380 CHNUM=144
1390 PLCH=0
1400 REM
1410 REM ------------------
1420 REM INITIIERUNGEN FUER
1430 REM DIE PLOT-FUNKTION
1440 REM ------------------
1450 REM
1460 F=2*PI/10
1470 XANFANG=0
1480 XENDE=10
1490 XDELTA=0.25
1500 REM
1510 REM -----------------
1520 REM BEARBEITUNGS-
1530 REM SCHLEIFE
1540 REM -----------------
1550 REM
1560 FOR LOOP=XANFANG TO XENDE STEP XDELTA
1570 REM
1580 X=LOOP :: Y=FKT(X)
1590 IXCH=(X-XA)*DXCHR
1600 IYCH=(Y-YA)*DYCHR
1610 ITEST=INT(IXCH)
1620 IF IXCH=ITEST THEN IXCH=ITEST-1 ELSE IXCH=ITEST
1630 ITEST=INT(IYCH)
1640 IF IYCH=ITEST THEN IYCH=ITEST-1 ELSE IYCH=ITEST
1650 IF IXCH=-1 THEN IXCH=0
1660 IF IYCH=-1 THEN IYCH=0
1670 IX=INT((X-(XA+IXCH*DXCH))*DXR)
1680 IY=INT((Y-(YA+IYCH*DYCH))*DYR)
1690 IY=9-IY
1700 IF IX=0 THEN IX=1
1710 IF IY=9 THEN IY=8
1720 V$=L$(IXCH,IYCH)
1730 REM
1740 REM
1750 REM
1760 IF LEN(V$)>0 THEN CALL ODERN(V$,IX,IY):: GOTO 1820
1770 REM
1780 REM
1790 IF IX>4 THEN P$="0"&STR$(2^(8-IX))
             ELSE P$=STR$(2^(4-IX))&"0"
```

```
1800 V$=RPT$("00",IY-1)&P$&RPT$("00",8-IY)
1810 PLCH=PLCH+1
1820 L$(IXCH,IYCH)=V$
1830 NEXT LOOP
1840 REM
1850 REM ----------------
1860 REM
1870 DISPLAY AT(24,5):"PLCH = ";PLCH
1880 FOR X=1 TO 25
1890 FOR Y=1 TO 20
1900 V$=L$(X,Y)
1910 IF LEN(V$)=0 THEN 1960
1920 SPALTE=X+4 :: ZEILE=22-Y
1930 CHNUM=CHNUM-1
1940 CALL CHAR(CHNUM,V$)
1950 CALL HCHAR(ZEILE,SPALTE,CHNUM,1)
1960 NEXT Y
1970 NEXT X
1980 ACCEPT AT(1,1):E$
1990 STOP
2000 REM
2010 REM
2020 REM ****************
2030 REM
2040 REM ================
2050 REM
2060 SUB CHTOHEX(CH$,HEX)
2070 REM
2080 REM ================
2090 REM
2100 REM
2110 IF CH$<="9" THEN HEX=ASC(CH$)-48 ELSE HEX=ASC(CH$)-55
2120 SUBEND
2130 REM
2140 REM ****************
2150 REM
2160 REM ================
2170 REM
2180 SUB HEXTOCH(HEX,CH$)
2190 REM
2200 REM ================
2210 REM
2220 REM
2230 IF HEX<=9 THEN CH$=CHR$(HEX+48)ELSE CH$=CHR$(HEX+55)
2240 SUBEND
2250 REM
2260 REM ****************
2270 REM
2280 REM ================
2290 REM
2300 SUB ODERN(V$,IX,IY)
2310 REM
2320 REM ================
2330 REM
2340 REM
2350 NZ=IY+IY
2360 IF IX<5 THEN NZ=NZ-1
2370 Z$=SEG$(V$,NZ,1)
2380 CALL CHTOHEX(Z$,Z)
```

```
2390 IF IX>4 THEN X=2^(8-IX)ELSE X=2^(4-IX)
2400 Z=Z OR X
2410 CALL HEXTOCH(Z,Z$)
2420 V$=SEG$(V$,1,NZ-1)&Z$&SEG$(V$,NZ+1,16-NZ)
2430 SUBEND
2440 REM
2450 REM
2460 REM ****************
```

Abb. 4.10: Plot-Programm für hochauflösende Grafik

PROGRAMME FÜR PRIVATE ANWENDUNGEN

In diesem Abschnitt wollen wir Ihnen einige Programme vorstellen, die besonders für den privaten Bereich sehr nützlich sind. Gerade für dieses Gebiet ist auf den verschiedenen Mikrocomputern eine Vielzahl von Programmen in BASIC geschrieben worden, so daß eine Auswahl schwerfällt. Wir haben einige Programme aus den Bereichen

- Prozentrechnung,
- Zinsrechnung,
- Tabellendarstellung,
- Kalkulation und
- Investitionsrechnung

ausgewählt, für den TI-99/4A umgeschrieben und zum Teil erweitert.

Da wir alle dargestellten Programme unter einem Hauptmenü zusammenfassen wollen, sind wir zunächst von einem neutralen Menü ausgegangen, in das wir schrittweise die einzelnen Teilprogramme eingebunden haben.

Neutrales Hauptmenü

Entsprechend unserer Forderung nach einer Strukturierung des Programm-Erstellungsprozesses und der Einzelprogramme selbst wollen wir das Programm mit dem Hauptmenü (Rumpfprogramm) so vollständig erstellen, daß bereits alle Programmfunktionen angesprochen werden können. Im einzelnen sind hierzu Routinen erforderlich für

- den Aufruf der einzelnen Programmfunktionen,
- das Einblenden von Überschriften bei Aufruf einer neuen Funktion,

– die Steuerung des Bildschirms,
– das Abfangen von falschen Ziffern bei der Funktionsauswahl und
– das Ein- und Ausschalten eines eventuell vorhandenen Druckers.

Aus dem Listing in Abb. 4.11 erkennen wir, daß bestimmten Zeilennummernbereichen auch ganz bestimmte Funktionen zugeordnet sind.

```
2 REM --------------------
4 REM    BASISMENU FUER DIE
6 REM  ERSTELLUNG BELIEBIGER
8 REM          PROGRAMME
10 REM --------------------
15 ON WARNING NEXT
20 D$=" "
25 HL$="----------------------------"
30 CALL CLEAR
40 DISPLAY AT(1,1):HL$
50 DISPLAY AT(2,1):"P0   - PROGRAMM 0    12.11.83"
60 DISPLAY AT(3,1):HL$
70 DISPLAY AT(6,1):"1 - FUNKTION 1"
80 DISPLAY AT(8,1):"2 - FUNKTION 2"
90 DISPLAY AT(10,1):"3 - FUNKTION 3"
100 DISPLAY AT(12,1):"8 - DRUCKER EIN/AUS ";D$
145 DISPLAY AT(15,1):"9 - ABBRUCH"
148 DISPLAY AT(18,1):"FUNKTION"
150 ACCEPT AT(18,14)VALIDATE("12389")BEEP SIZE(1):M
300 IF M=1 THEN GOSUB 1000
310 IF M=2 THEN GOSUB 2000
320 IF M=3 THEN GOSUB 3000
330 IF M=8 THEN GOSUB 8000
340 IF M=9 THEN GOTO 500
400 GOTO 30
500 CALL CLEAR
510 END
1000 REM ********************
1010 REM FUNKTION 1
1020 REM ********************
1050 REM BEGIN
1060 GOSUB 1900
1800 GOSUB 9100
1810 IF W$="J" THEN GOTO 1000
1890 RETURN
1900 REM ********************
1905 REM KOPF 1
1907 REM ********************
1908 CALL CLEAR
1910 DISPLAY AT(1,1):HL$
1915 DISPLAY AT(2,1):"P01    -    PROGRAMMFUNKTION 1"
1920 DISPLAY AT(3,1):HL$
1990 RETURN
2000 REM ********************
2010 REM FUNKTION 2
2020 REM ********************
2060 GOSUB 2900
2800 GOSUB 9100
```

```
2810 IF W$="J" THEN GOTO 2000
2890 RETURN
2900 REM *******************
2905 REM KOPF 2
2908 REM *******************
2909 CALL CLEAR
2910 DISPLAY AT(1,1):HL$
2915 DISPLAY AT(2,1):"P02   -    PROGRAMMFUNKTION 2"
2920 DISPLAY AT(3,1):HL$
2990 RETURN
3000 REM *******************
3005 REM FUNKTION 3
3010 REM *******************
3060 GOSUB 3900
3800 GOSUB 9100
3810 IF W$="J" THEN GOTO 3000
3890 RETURN
3900 REM *******************
3905 REM KOPF 3
3907 REM *******************
3909 CALL CLEAR
3910 DISPLAY AT(1,1):HL$
3915 DISPLAY AT(2,1):"P03   -    PROGRAMMFUNKTION 3"
3920 DISPLAY AT(3,1):HL$
3990 RETURN
8000 REM *******************
8005 REM DRUCKER EIN/AUS
8008 REM *******************
8010 IF D$=" " THEN D$="*" :: RETURN
8020 IF D$="*" THEN D$=" " :: RETURN
8030 REM *******************
9000 REM DIV. ROUTINEN
9010 REM *******************
9100 REM UNTERROUTINE ABBRUCH
9110 REM *******************
9120 DISPLAY AT(23,1):"WEITER, DANN J EINGEBEN"
9130 ACCEPT AT(23,25)VALIDATE("JN")SIZE(1):W$
9140 RETURN
9150 REM *******************
9200 REM UNTERROUTINE FORTSETZUNG
9210 REM *******************
9250 DISPLAY AT(23,1):"FORTS., DANN X EING."
9260 ACCEPT AT(23,22)SIZE(1):N$
9270 IF N$="X" THEN RETURN
9280 GOTO 10
9290 REM *******************
```

Abb. 4.11: Programm zur Erstellung eines Basismenüs für beliebige Programme

Der Hauptsteuerbereich bildet den Rumpf des Programms. Hier wird das Startmenü aufbereitet und auf die einzelnen Programmfunktionen verzweigt (1000, 2000, 3000). Fehleingaben werden durch die VALIDATE- und SIZE-Anweisung abgefangen und durch die BEEP-Anweisung akustisch angezeigt.

```
2 REM --------------------
4 REM    BASISMENU FUER DIE
6 REM  ERSTELLUNG BELIEBIGER
8 REM          PROGRAMME
10 REM --------------------
15 ON WARNING NEXT
20 D$=" "
25 HL$="----------------------------"
30 CALL CLEAR
40 DISPLAY AT(1,1):HL$
50 DISPLAY AT(2,1):"P0   -   PROGRAMM 0    12.11.83"
60 DISPLAY AT(3,1):HL$
70 DISPLAY AT(6,1):"1 - FUNKTION 1"
80 DISPLAY AT(8,1):"2 - FUNKTION 2"
90 DISPLAY AT(10,1):"3 - FUNKTION 3"
100 DISPLAY AT(12,1):"8 - DRUCKER EIN/AUS ";D$
145 DISPLAY AT(15,1):"9 - ABBRUCH"
148 DISPLAY AT(18,1):"FUNKTION"
150 ACCEPT AT(18,14)VALIDATE("12389")BEEP SIZE(1):M
300 IF M=1 THEN GOSUB 1000
310 IF M=2 THEN GOSUB 2000
320 IF M=3 THEN GOSUB 3000
330 IF M=8 THEN GOSUB 8000
340 IF M=9 THEN GOTO 500
400 GOTO 30
500 CALL CLEAR
510 END
```

Abb. 4.12: Hauptsteuerbereich des Programms aus Abb. 4.11

Entsprechend der Eingabe auf die Anfrage „FUNKTION“ wird zu den einzelnen Programmfunktionen mit einer GOSUB-Anweisung verzweigt (1000 bei 1, 2000 bei 2, 3000 bei 3, 8000 bei 8 und 500 bei 9).

Innerhalb der Programmfunktionen sind jeweils die Bereiche von

- x000 bis x060 für den Aufruf der Funktionsüberschriften (z.B. 1000 bis 1060),
- x800 bis x890 für die Entscheidung einer erneuten Durchführung der beendeten Programmfunktion (z.B. 1800 bis 1890) und
- x900 bis x990 für die Anzeige der Funktionsüberschrift (z.B. 1900 bis 1990)

festgelegt.

Hierdurch stehen für eigene Programmanweisungen jeweils die Bereiche von x100 bis x790 (z.B. von 1100 bis 1790) zur Verfügung.

Mit der Programmfunktion „8“ wird im Hauptmenü ein Schalter für die

Anzeige eines parallel geschalteten Druckers gesetzt. Wenn Sie über einen Drucker verfügen, so kann der Schalter innerhalb der Programmfunktionen für die Druckersteuerung verwendet werden. Wir haben im Rahmen der hier gezeigten Programme darauf bewußt verzichtet.

Für die Steuerung der innerhalb der Programme im allgemeinen vorkommenden Bildschirmanzeigen und die Behandlung einer abgearbeiteten Programmfunktion haben wir zwei Unterroutinen geschrieben.

Unterroutine ABBRUCH

Die Unterroutine ABBRUCH soll bei einer Beantwortung mit „J“ die beendete Programmfunktion erneut starten. Es wird dann nach dem Rücksprung hinter die Aufrufanweisung an den Beginn der entsprechenden Programmfunktion gesprungen (z.B. 1810 IF W$ = „J“ THEN GOTO 1000).

Unterroutine FORTSETZUNG

Die Unterroutine FORTSETZUNG soll, wenn „X“ geantwortet wurde, mit der Programmausführung fortfahren. In dem hier gezeigten Programm (Abb. 4.11) wird sie nicht ausgeführt. Sie wurde für spätere Anwendungen in dem noch offenen Bereich x100 bis x890 mitaufgeführt. Sie können sie dort entsprechend einsetzen (vgl. Anwendung in den folgenden Programmen).

Wir empfehlen Ihnen, vor Eingabe der anderen Programme dieses Programm einmal vollständig einzugeben und auszutesten. Versuchen Sie dabei auch gleichzeitig, die Zeilennummern so zu wählen, daß die vorgegebenen Grenzen nicht überschritten und daß jeweils gleiche Zeilenabstände erreicht werden, z.B.:

- 10 bis maximal 95 für die Basisdefinitionen, 100 bis maximal 900 für die Hauptanzeige,
- x000 bis maximal x090 für die jeweiligen Vorbereiche der einzelnen Programmfunktionen
- und so weiter

entsprechend der Vorlage des Programms in Abb. 4.11.

Einfache Prozentrechnungen

In diesem Programm wollen wir in zwei Programmfunktionen einfache Prozentrechnungen durchführen. Wir haben hierzu einmal eine Funk-

tion für das Nachrechnen oder auch Erstellen von Rechnungen und zum anderen eine Funktion für Prozentkettenrechnungen ausgewählt.



Abb. 4.13: Struktur und Menü der Prozentrechnungsprogramme

```
10 ON WARNING NEXT
15 D$=" "
20 HL$="----------------------------"
30 CALL CLEAR
40 DISPLAY AT(1,1):HL$
50 DISPLAY AT(2,1):"P1 - EINF. PROZENTRECHNUNGEN"
60 DISPLAY AT(3,1):HL$
70 DISPLAY AT(6,1):"1 - MWST, RABATT, SKONTO"
80 DISPLAY AT(8,1):"2 - ALLG. PROZENTRECHNUNG"
100 DISPLAY AT(10,1):"8 - (DRUCKER EIN/AUS) ";D$
145 DISPLAY AT(15,1):"9 - ABBRUCH"
148 DISPLAY AT(18,1):"FUNKTION"
150 ACCEPT AT(18,14)VALIDATE("1289")BEEP SIZE(1):M
300 IF M=1 THEN GOSUB 1000
310 IF M=2 THEN GOSUB 2000
330 IF M=8 THEN GOSUB 8000
340 IF M=9 THEN GOTO 500
400 GOTO 30
500 CALL CLEAR
510 END
1000 REM ********************
1010 REM MWST, RABATT, SKONTO
1020 REM ********************
1050 REM BEGIN
1060 GOSUB 1900
1070 DISPLAY AT(18,1):"EINGABE VON 0 BEI DER GE-"
1080 DISPLAY AT(19,1):"GROESSE (GRUNDBETRAG ODER"
1085 DISPLAY AT(20,1):"ENDBETRAG)"
1100 DISPLAY AT(5,1):"GRUNDBETRAG (DM)"
1105 ACCEPT AT(5,20)VALIDATE(NUMERIC):A
1110 DISPLAY AT(6,1):"RABATT (%)"
1115 ACCEPT AT(6,20)VALIDATE(NUMERIC):B
1120 DISPLAY AT(7,1):"MEHRWERTSTEUER (%)"
1125 ACCEPT AT(7,20)VALIDATE(NUMERIC):C
```

```
1130 DISPLAY AT(8,1):"SKONTO (%)"
1135 ACCEPT AT(8,20)VALIDATE(NUMERIC):D
1140 DISPLAY AT(9,1):"ENDBETRAG (DM)"
1145 ACCEPT AT(9,20)VALIDATE(NUMERIC):E
1150 IF A<>0 THEN GOSUB 1300
1155 IF A=0 THEN GOSUB 1200
1165 GOTO 1800
1200 REM ENDBETRAGSRECHNUNG
1205 Z2=E/(1-D/100)
1210 DB=Z2-E
1215 CB=Z2-Z2/(1+C/100)
1220 Z1=Z2-CB
1225 BB=Z1/(1-B/100)-Z1
1230 A=Z1+BB
1250 GOSUB 1400
1290 RETURN
1295 REM ********************
1300 REM ENDBETRAGSRECHNUNG VOM ANFANG
1305 BB=A*B/100
1310 Z1=A-BB
1315 CB=Z1*C/100
1320 Z2=Z1+CB
1325 DB=Z2*D/100
1330 E=Z2-DB
1350 GOSUB 1400
1390 RETURN
1395 REM ********************
1400 REM DRUCK
1402 REM ********************
1405 GOSUB 1900
1410 DISPLAY AT(7,1):"GRUNDBETRAG       DM"
1415 DISPLAY AT(7,20):USING "######.##":A
1420 DISPLAY AT(8,1):"RABATT            DM"
1425 DISPLAY AT(8,20):USING "######.##":-BB
1430 DISPLAY AT(9,1):"ZWISCHENSUMME 1   DM"
1435 DISPLAY AT(9,20):USING "######.##":Z1
1440 DISPLAY AT(10,1):"MWST             DM"
1445 DISPLAY AT(10,20):USING "######.##":CB
1450 DISPLAY AT(11,1):"ZWISCHENSUMME 2  DM"
1455 DISPLAY AT(11,20):USING "######.##":Z2
1460 DISPLAY AT(12,1):"SKONTO           DM"
1465 DISPLAY AT(12,20):USING "######.##":-DB
1470 DISPLAY AT(13,1):"                   -----------"
1480 DISPLAY AT(14,1):"ENDBETRAG"
1485 DISPLAY AT(14,20):USING "######.##":E
1490 RETURN
1495 REM ********************
1800 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
1805 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
1810 IF W$="J" THEN GOTO 1050
1890 RETURN
1895 REM ********************
1900 REM KOPF 2
1903 REM ********************
1905 CALL CLEAR
1910 DISPLAY AT(1,1):HL$
1915 DISPLAY AT(2,1):"P11  -  MWST, RABATT, SKONTO"
1920 DISPLAY AT(3,1):HL$
1990 RETURN
```

```
2000 REM ********************
2010 REM ALLGEMEINE PROZENTRECHNUNG
2020 REM ********************
2050 REM BEGIN
2060 GOSUB 2900
2100 DISPLAY AT(5,1):"AUSGANGSBETRAG  DM"
2105 ACCEPT AT(5,20)VALIDATE(NUMERIC):A
2110 Z=A
2115 FOR I=1 TO 3
2120 DISPLAY AT(I*4+4,1):"AUF-/ABSCHLAG   %"
2125 ACCEPT AT(I*4+4,20)VALIDATE(NUMERIC):B
2128 BB=Z*B/100
2129 DISPLAY AT(I*4+5,1):"AUF-/ABSCHLAG   DM"
2130 DISPLAY AT(I*4+5,20):USING "######.##":BB
2135 Z=Z+BB
2138 DISPLAY AT(I*4+6,1):"ZWISCHENSUMME   DM"
2140 DISPLAY AT(I*4+6,20):USING "######.##":Z
2190 NEXT I
2800 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
2805 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
2810 IF W$="J" THEN GOTO 2050
2890 RETURN
2895 REM ********************
2900 REM KOPF 2
2905 REM ********************
2906 CALL CLEAR
2910 DISPLAY AT(1,1):HL$
2915 DISPLAY AT(2,1):"P12    -    ALLG. PROZENTRCHN."
2920 DISPLAY AT(3,1):HL$
2990 RETURN
2995 REM ********************
8000 REM DRUCKER EIN/AUS
8005 REM ********************
8010 IF D$=" " THEN D$="*" :: RETURN
8020 IF D$="*" THEN D$=" " :: RETURN
8900 REM ********************
```

Abb. 4.14: Prozentrechnungs-Programm

Mehrwertsteuer, Rabatt, Skonto

Dieses Programm basiert auf dem einfachen Zusammenhang

ENDBETRAG =
NETTOBETRAG – RABATT + MWST – SKONTO

Hierbei werden zwei Abläufe behandelt. Im ersten Ablauf wird der Endbetrag ermittelt. Er wird immer dann ausgeführt, wenn der Grundbetrag ungleich 0 angegeben wird. Im zweiten Ablauf wird der Grundbetrag in einer Rückwärtsrechnung ermittelt. Dieser Ablauf wird gestartet, wenn bei der Abfrage nach dem Grundbetrag mit 0 geantwortet wird.

```
1200 REM ENDBETRAGSRECHNUNG
1205 Z2=E/(1-D/100)
1210 DB=Z2-E
1215 CB=Z2-Z2/(1+C/100)
1220 Z1=Z2-CB
1225 BB=Z1/(1-B/100)-Z1
1230 A=Z1+BB
1250 GOSUB 1400
1290 RETURN
```

Abb. 4.15: Unterprogramm Endbetragsrechnung

```
1300 REM ENDBETRAGSRECHNUNG VOM ANFANG
1305 BB=A*B/100
1310 Z1=A-BB
1315 CB=Z1*C/100
1320 Z2=Z1+CB
1325 DB=Z2*D/100
1330 E=Z2-DB
1350 GOSUB 1400
1390 RETURN
```

Abb. 4.16: Unterprogramm Grundbetragsrechnung

Damit bei der Dateneingabe keine unsinnigen Zahlen eingegeben werden (z.B. Rabatt > 90 %, MWSt > 30 % und Skonto > 15 %), werden die einzelnen Eingaben direkt nach ihrer Erfassung überprüft. Bei Überschreiten der vorgegebenen Grenzen wird die Erfassung erneut durchgeführt. Prüfen Sie für Ihre Anwendung, ob weitere Einschränkungen möglich sind.

```
1070 DISPLAY AT(18,1):"EINGABE VON 0 BEI DER GE-"
1080 DISPLAY AT(19,1):"GROESSE (GRUNDBETRAG ODER"
1085 DISPLAY AT(20,1):"ENDBETRAG)"
1100 DISPLAY AT(5,1):"GRUNDBETRAG (DM)"
1105 ACCEPT AT(5,20)VALIDATE(NUMERIC):A
1110 DISPLAY AT(6,1):"RABATT (%)"
1115 ACCEPT AT(6,20)VALIDATE(NUMERIC):B
1120 DISPLAY AT(7,1):"MEHRWERTSTEUER (%)"
1125 ACCEPT AT(7,20)VALIDATE(NUMERIC):C
1130 DISPLAY AT(8,1):"SKONTO (%)"
1135 ACCEPT AT(8,20)VALIDATE(NUMERIC):D
1140 DISPLAY AT(9,1):"ENDBETRAG (DM)"
1145 ACCEPT AT(9,20)VALIDATE(NUMERIC):E
```

Abb. 4.17: Programmteil Datenerfassung

Allgemeine Prozentrechnung

Wir wollen nun die Prozentrechnung erweitern. Hierzu legen wir folgende Vorschrift fest. Ausgehend von einem Grundbetrag, werden Prozentsätze eingegeben, die je nach Vorzeichen addiert oder subtrahiert werden. Insgesamt können nach der Eingabe des Grundbetrags innerhalb eines Bildschirmbereichs drei Zu- und/oder Abschläge eingegeben werden. Nach Erreichung von drei Eingaben wird abgefragt, ob weiter gerechnet werden soll. Bei Beantwortung mit „J" können wiederum drei Eingaben erfolgen.

```
2100 DISPLAY AT(5,1):"AUSGANGSBETRAG  DM"
2105 ACCEPT AT(5,20)VALIDATE(NUMERIC):A
2110 Z=A
2115 FOR I=1 TO 3
2120 DISPLAY AT(I*4+4,1):"AUF-/ABSCHLAG   %"
2125 ACCEPT AT(I*4+4,20)VALIDATE(NUMERIC):B
2128 BB=Z*B/100
2129 DISPLAY AT(I*4+5,1):"AUF-/ABSCHLAG   DM"
2130 DISPLAY AT(I*4+5,20):USING "######.##":BB
2135 Z=Z+BB
2138 DISPLAY AT(I*4+6,1):"ZWISCHENSUMME   DM"
2140 DISPLAY AT(I*4+6,20):USING "######.##":Z
2190 NEXT I
2800 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
2805 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
2810 IF W$="J" THEN GOTO 2050
2890 RETURN
```

Abb. 4.18: Unterprogramm Erfassungs- und Anzeigebereich

Sie erkennen, daß die Beträge auf zwei Stellen hinter dem Komma abgeschnitten werden (z.B. Zeile 2130). Für dieses Programm bieten sich insbesondere folgende Erweiterungen an:

- Auf- oder Abrunden der Beträge,
- Eingrenzen der möglichen prozentuellen Zu- und Abschläge auf Prozentzahlen unter 100 % (Normalfall),
- Eingabe der Zu- und Abschläge als Absolutbeträge und Errechnung der Prozentsätze. Als Erkennungszeichen für das Programm könnte z.B. bei den Eingaben, die als Prozentbeträge interpretiert werden sollen, an die Zahl das Zeichen % angehängt werden.

Weitere Programme zum Bereich Prozentrechnungen

Innerhalb des Prozentrechnungs-Programms (Abb. 4.14) sind unter Beibehaltung der Grundstruktur weitere 5 Programmfunktionen (3–7) möglich. Wenn Sie nach weiteren Anwendungen suchen, denken Sie z.B. auch an die verschiedenen Arten von Dreisatzrechnungen. Sie können dann für alle Funktionen die gleiche Grundstruktur wie bei den Funktionen 1 und 2 verwenden.

ZINSRECHNUNGEN

In diesem Abschnitt beschreiben wir einige Zinsrechnungsprogramme. Es sind dies im einzelnen Programme der allgemeinen Zinsrechnung, der Schuldzinsrechnung und der Zinsstaffelrechnung.

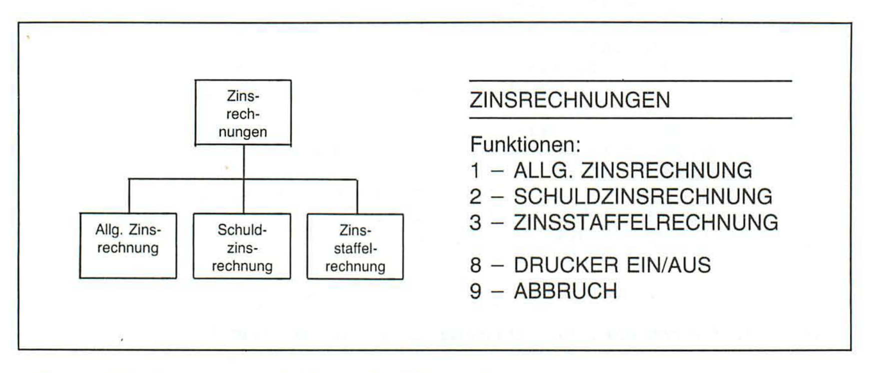


Abb. 4.19: Struktur und Menü der Zinsrechnungsprogramme

```
2 REM --------------------
4 REM     EINFACHE ZINSRECH-
6 REM       NUNGSPROGRAMME
10 REM -------------------
15 ON WARNING NEXT
20 D$=" "
25 HL$="----------------------------"
30 CALL CLEAR
40 DISPLAY AT(1,1):HL$
50 DISPLAY AT(2,1):"P2  -    EINF. ZINSRECHNUNGEN"
60 DISPLAY AT(3,1):HL$
70 DISPLAY AT(6,1):"1 - ALLG. ZINSRECHNUNG"
80 DISPLAY AT(8,1):"2 - SCHULDZINSRECHNUNG"
90 DISPLAY AT(10,1):"3 - STAFFELZINSRECHNUNG"
```

```
100 DISPLAY AT(12,1):"8 - DRUCKER EIN/AUS ";D$
145 DISPLAY AT(15,1):"9 - ABBRUCH"
148 DISPLAY AT(18,1):"FUNKTION"
150 ACCEPT AT(18,14)VALIDATE("12389")BEEP SIZE(1):M
300 IF M=1 THEN GOSUB 1000
310 IF M=2 THEN GOSUB 2000
320 IF M=3 THEN GOSUB 3000
330 IF M=8 THEN GOSUB 8000
335 IF M=9 THEN GOTO 500
340 DISPLAY AT(9,20):USING ("######.##"):N
400 GOTO 30
500 CALL CLEAR
510 END
1000 REM *******************
1010 REM EINFACHE ZINSRECHNUNG
1020 REM *******************
1050 REM BEGIN
1060 GOSUB 1900
1070 DISPLAY AT(16,1):"BITTE GEBEN SIE BEI DER"
1075 DISPLAY AT(17,1):"GROESSE DIE ERRECHNET WERDEN"
1080 DISPLAY AT(18,1):"SOLL, EINE 0 EIN"
1100 DISPLAY AT(5,1):"ANFANGSKAPITAL"
1120 ACCEPT AT(5,22)VALIDATE(NUMERIC)SIZE(8):A
1130 DISPLAY AT(6,1):"ZINSSATZ"
1140 ACCEPT AT(6,22)VALIDATE(NUMERIC)SIZE(5):B
1145 IF B>45 THEN GOTO 1140
1150 DISPLAY AT(7,1):"ANZAHL DER JAHRE"
1160 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(3):C
1165 IF C>99 THEN GOTO 1160
1170 DISPLAY AT(8,1):"ENDKAPITAL"
1180 ACCEPT AT(8,22)VALIDATE(NUMERIC)SIZE(8):D
1190 DISPLAY AT(9,1):"ZINSTERMINE/JAHR"
1200 ACCEPT AT(9,22)VALIDATE(NUMERIC)SIZE(8):E
1210 REM
1220 IF E=0 THEN E=1
1230 I=C*E :: B=B/E
1250 IF A=0 THEN A=D/(1+B/100)^I :: GOTO 1520
1260 IF B=0 THEN B=((D/A)^(1/I)-1)*100*E :: GOTO 1540
1270 IF C=0 THEN C=LOG(D/A)/LOG(1+B/100)/E :: GOTO 1560
1280 IF D=0 THEN D=A*(1+B/100)^I :: GOTO 1580
1290 GOTO 1100
1500 REM ANZEIGBEREICH
1520 DISPLAY AT(12,1):"BENOET. ANF.KAP."
1530 DISPLAY AT(12,20):USING ("######.##"):A :: GOTO 1600
1540 DISPLAY AT(12,1):"BENOET. ZINSFUSS"
1550 DISPLAY AT(12,20):USING ("######.##"):B :: GOTO 1600
1560 DISPLAY AT(12,1):"ANZAHL DER JAHRE"
1570 DISPLAY AT(12,20):USING ("######.##"):C :: GOTO 1600
1580 DISPLAY AT(12,1):"ENDKAPITAL"
1590 DISPLAY AT(12,20):USING ("######.##"):D :: GOTO 1600
1600 REM
1800 GOSUB 9100
1810 IF W$="J" THEN GOTO 1000
1890 RETURN
1900 REM *******************
1905 REM KOPF 2
1907 REM *******************
1908 CALL CLEAR
1910 DISPLAY AT(1,1):HL$
1915 DISPLAY AT(2,1):"P21    -    ALLG. ZINSRECHNUNG"
```

```
1920 DISPLAY AT(3,1):HL$
1990 RETURN
2000 REM *******************
2010 REM SCHULDZINSRECHNUNG
2020 REM *******************
2030 F=0 :: G=0
2100 REM DATENERFASSUNG
2110 GOSUB 2900
2120 DISPLAY AT(5,1):"ANZAHL KAPITALIEN"
2130 ACCEPT AT(5,22)VALIDATE(DIGIT)SIZE(1):A
2135 IF A=0 THEN GOTO 2130
2140 FOR I=1 TO A
2150 GOSUB 2900
2160 DISPLAY AT(5,1):"BERECHNUNG FUER KAPITAL"
2170 DISPLAY AT(5,24):I
2180 DISPLAY AT(7,1):"ZINSEN IN %"
2190 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(5):B
2200 DISPLAY AT(8,1):"KAPITAL"
2210 ACCEPT AT(8,22)VALIDATE(NUMERIC)SIZE(8):C
2220 DISPLAY AT(9,1):"ANZAHL DER TAGE"
2230 ACCEPT AT(9,22)VALIDATE(DIGIT)SIZE(3):D
2240 E=C*B/100*D/365
2250 G=G+E :: F=F+C
2300 DISPLAY AT(11,1):"DIES ERGIBT FUER KAPITAL"
2310 DISPLAY AT(11,25):I
2320 DISPLAY AT(13,1):"ZINSBETRAG"
2330 DISPLAY AT(13,20):USING ("######.##"):E
2340 DISPLAY AT(14,1):"SUMME KAPITAL"
2350 DISPLAY AT(14,20):USING ("######.##"):C+E
2360 GOSUB 9200
2380 NEXT I
2400 DISPLAY AT(16,1):"GES.SUMME ZINSEN"
2410 DISPLAY AT(16,20):USING ("######.##"):G
2420 DISPLAY AT(17,1):"GES.SUMME KAPITAL"
2430 DISPLAY AT(17,20):USING ("######.##"):F
2440 DISPLAY AT(18,1):"                   ----------"
2450 DISPLAY AT(19,20):USING ("######.##"):F+G
2490 GOSUB 9100
2495 IF W$="J" THEN GOTO 2000
2500 RETURN
2800 GOSUB 9100
2810 IF W$="J" THEN GOTO 2000
2890 RETURN
2900 REM *******************
2905 REM KOPF 3
2908 REM *******************
2909 CALL CLEAR
2910 DISPLAY AT(1,1):HL$
2915 DISPLAY AT(2,1):"P22  -  SCHULDZINSBERECHNUNG"
2920 DISPLAY AT(3,1):HL$
2990 RETURN
3000 REM *******************
3005 REM FUNKTION 3
3010 REM *******************
3060 GOSUB 3900
3100 REM DATENERFASSUNG
3110 DISPLAY AT(5,1):"UEBERZ.-GRENZE"
3120 ACCEPT AT(5,22)VALIDATE(NUMERIC)SIZE(8):A
3130 DISPLAY AT(6,1):"ZINSSATZ"
3140 ACCEPT AT(6,22)VALIDATE(NUMERIC)SIZE(5):D
```

```
3150 DISPLAY AT(7,1):"KREDITPROV."
3160 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(5):E
3170 DISPLAY AT(8,1):"UEBERZ.PROV."
3180 ACCEPT AT(8,22)VALIDATE(NUMERIC)SIZE(5):F
3190 DISPLAY AT(9,1):"SALDO"
3200 ACCEPT AT(9,22)VALIDATE(NUMERIC)SIZE(8):B
3210 DISPLAY AT(10,1):"ANZAHL DER TAGE"
3220 ACCEPT AT(10,22)VALIDATE(DIGIT)SIZE(3):C
3250 G=INT(B*C*D/360+0.5)/100
3260 IF B>A THEN H=INT((B-A)*F*C/360+0.5)/100
3270 I=I+G :: J=J+H :: K=K+C :: H=0
3300 REM WEITER
3310 M=(A*K/100+J*360/F-I*360/D)/(360/E)
3320 GOSUB 3900
3330 DISPLAY AT(5,1):"SOLLZINSEN"
3340 DISPLAY AT(5,20):USING ("######.##"):I
3350 DISPLAY AT(6,1):"UEBERZ.PROV."
3360 DISPLAY AT(6,20):USING ("######.##"):J
3370 DISPLAY AT(7,1):"KREDITPROV."
3380 DISPLAY AT(7,20):USING ("######.##"):M
3385 N=B+I+J+M
3390 DISPLAY AT(9,1):"NEUER SALDO"
3400 DISPLAY AT(9,20):USING ("######.##"):N
3500 DISPLAY AT(22,1):"NEUER SALDO, DANN J"
3510 ACCEPT AT(22,22)VALIDATE("JN")SIZE(1):S$
3520 IF S$="J" THEN GOSUB 3900 :: GOTO 3190
3800 GOSUB 9100
3810 IF W$="J" THEN GOTO 3000
3890 RETURN
3900 REM ********************
3905 REM KOPF 3
3907 REM ********************
3909 CALL CLEAR
3910 DISPLAY AT(1,1):HL$
3915 DISPLAY AT(2,1):"P23    -   ZINSSTAFFELRECHNUNG"
3920 DISPLAY AT(3,1):HL$
3990 RETURN
8000 REM ********************
8005 REM DRUCKER EIN/AUS
8008 REM ********************
8010 IF D$=" " THEN D$="*" :: RETURN
8020 IF D$="*" THEN D$=" " :: RETURN
8030 REM ********************
9000 REM DIV. ROUTINEN
9010 REM ********************
9100 REM UNTERROUTINE ABBRUCH
9110 REM ********************
9120 DISPLAY AT(23,1):"WEITER, DANN J EINGEBEN"
9130 ACCEPT AT(23,25)VALIDATE("JN")SIZE(1):W$
9140 RETURN
9150 REM ********************
9200 REM UNTERROUTINE FORTSETZUNG
9210 REM ********************
9250 DISPLAY AT(23,1):"FORTS., DANN X EING."
9260 ACCEPT AT(23,22)SIZE(1):N$
9270 IF N$="X" THEN RETURN
9280 GOTO 10
9290 REM ********************
```

Abb. 4.20: Zinsrechnungs-Programm

Allgemeine Zinsrechnung

Mit diesem Programm sollen nach der allgemeinen Zinsformel die gewünschten offenen Werte errechnet werden. Hierzu ist im Eingabezyklus bei der Variablen, die errechnet werden soll, eine 0 einzugeben.

Allgemeine Zinsformel:

$$K_n = K_0 * \left(1 + \frac{p}{100}\right)^{n*i}$$

K_0 = Anfangskapital
p = Zinssatz in %
n = Anzahl der Jahre, die das Anfangskapital zum Zinssatz p verzinst werden soll
i = Anzahl der Zinszahlungen im Jahr (Im allgemeinen ist i = i. Der akkumulierte Zins erhöht hierdurch die für den nächsten Zeitraum zu verzinsende Kapitalsumme)
K_n = Endkapital nach n Jahren bei i Zinsauszahlungen im Jahr

Folgende Beispiele wollen wir gemeinsam durchrechnen:

Beispiel 1

Es steht ein Kapital von 10.000 DM zur Verfügung, und wir wollen wissen, wieviel Jahre bei einem mittleren Zinssatz von 7 % vergehen müssen, um dieses Anfangskapital auf das Endkapital von 20.000 DM zu verdoppeln (vgl. Rechenlauf 1).

```
P21 – ALLG. ZINSRECHNUNG

ANFANGSKAPITAL                10000
ZINSSATZ                      7
ANZAHL DER JAHRE              0
ENDKAPITAL                    20000
ZINSTERMINE JAHR              1

ANZAHL DER JAHRE              10.24

BITTE GEBEN SIE BEI DER GROESSE, DIE
BERECHNET WERDEN SOLL, EINE 0 EIN

WEITER, DANN J EINGEBEN
```

Beispiel 2

Wir wollen in 10 Jahren eine Gesamtsumme von 30.000 DM mit einem Zinssatz von 8¼% ansparen. Welcher Kapitaleinsatz ist hierzu bei jährlichen Zinszahlungen erforderlich (vgl. Rechenlauf 2)?

```
P21 – ALLG. ZINSRECHNUNG
ANFANGSKAPITAL              0
ZINSSATZ                    8.25
ANZAHL DER JAHRE            10
ENDKAPITAL                  30000
ZINSTERMINE JAHR            1

BENOET. ANF. KAP.           13578.20

BITTE GEBEN SIE BEI DER GROESSE, DIE
BERECHNET WERDEN SOLL, EINE 0 EIN

WEITER, DANN J EINGEBEN
```

Beispiel 3

Sie haben einen Betrag von 15.000 DM zur Verfügung. Hiermit wollen Sie sich in 4 Jahren ein neues Auto kaufen, für das Sie dann 22.000 DM bezahlen müssen. Welchen Zinssatz müssen Sie hierzu mit Ihrer Bank bei halbjährlichen Zinszahlungen aushandeln (vgl. Rechenlauf 3)?

```
P21 – ALLG. ZINSRECHNUNG
ANFANGSKAPITAL              15000
ZINSSATZ                    0
ANZAHL DER JAHRE            4
ENDKAPITAL                  22000
ZINSTERMINE JAHR            2

BENOET. ZINSFUSS            9.81

BITTE GEBEN SIE BEI DER GROESSE, DIE
BERECHNET WERDEN SOLL, EINE 0 EIN

WEITER, DANN J EINGEBEN
```

Beispiel 4

Sie erwerben eine Beteiligung von 20.000 DM zu einem garantierten Zinssatz von 9 % bei jährlichen Zinszahlungen. Wie hoch wird Ihre Beteiligung nach 10 Jahren sein (vgl. Rechenlauf 4)?

```
P21 – ALLG. ZINSRECHNUNG

ANFANGSKAPITAL          20000
ZINSSATZ                9
ANZAHL DER JAHRE        10
ENDKAPITAL              0
ZINSTERMINE JAHR        1

ENDKAPITAL              47347.27

BITTE GEBEN SIE BEI DER GROESSE, DIE
BERECHNET WERDEN SOLL, EINE 0 EIN

WEITER, DANN J EINGEBEN
```

Das Programm

Innerhalb des Programms wurde aus Vereinfachungsgründen bei den einzelnen Variablen kein Bezug zu den Formelgrößen hergestellt.

Als Hilfsgrößen im Programm werden verwendet:

$n * i = C * E$ = Hilfsgröße für die Gesamtzahl der Zinstermine
$p / i = B / E$ = Hilfsgröße für den Zinssatz während der Zinsperiode (Zeitraum zwischen den Zinszahlungen)

Erweiterung

Erweitern Sie das Programm durch die Möglichkeit eines Paralleldrucks mit Ihrem Drucker, der durch eine Zusatzangabe im Menü gestartet werden soll.

Schuldzinsrechnung

In diesem Programm berechnen wir die Zinsen (Zinserträge und -aufwendungen) für verschiedene Kapitalien mit jeweils unterschiedlichen Laufzeiten und veränderten Zinssätzen. Es sollen hierbei einmal für die

einzelnen Vermögenswerte die Zinsen und Endsummen und zum anderen die Gesamtzinsen und die Gesamtsumme aller Vermögenswerte ausgewiesen werden.

Zinsformeln:

$$Z_i = \frac{K_{0i} * T_d * p}{360 * 100}$$

K_i $= K_{0i} + Z_i$
Z_g =Summe aller Z_i
K_{ng} =Summe aller Endvermögenswerte
K_{0i} =Anfangskapital i
p_i =Zinssatz (%) für Kapital i
Z_i =Zinsen für Kapital i
K_{ni} =Endkapital aus Anfangskapital i und den hierfür angefallenen Zinsen
Z_g =Summe der Zinsen aus allen Vermögenswerten
K_{ng} =Summe aller Endvermögenswerte
T_d =Dauer in Tagen

Beispiel

Wir haben von einer Bank Darlehen in Höhe von 1.000 DM und 2.000 DM in Anspruch genommen. Das erste Darlehen läuft über einen Zeitraum von 135 Tagen mit einem Zinssatz von 10 %, und das zweite Darlehen hat eine Laufzeit von 200 Tagen mit einem Zinssatz von 5 %. Für die Rückzahlung wollen wir die notwendigen Werte errechnen.

```
P22 – SCHULDZINSBERECHNUNG

BERECHNUNG FUER KAPITAL 1

ZINSEN IN %                    10
KAPITAL                        1000
ANZAHL DER TAGE                135

DIES ERGIBT FUER KAPITAL 1

ZINSBETRAG                       36.99
SUMME KAPITAL                  1036.099

FORTS., DANN X EING.           X
```

P22 – SCHULDZINSBERECHNUNG	
ZINSEN IN %	5
KAPITAL	2000
ANZAHL DER TAGE	200
DIES ERGIBT FUER KAPITAL 2	
ZINSBETRAG	131.51
SUMME KAPITAL	2131.51
GES. SUMME ZINSEN	168.49
GES. SUMME KAPITAL	3000.00
	3168.49
WEITER, DANN J EINGEBEN	

Zinsstaffelrechnung

Mit dem Programm der Zinsstaffelrechnung wollen wir Staffelrechnungen für Kontokorrentkredite durchführen, wie sie beispielsweise bei Ihrem Girokonto vorkommen können. Hierbei werden sowohl Kredit- als auch Überziehungsprovisionen mit berücksichtigt. Die Zinsstaffelrechnung wird dabei so lange fortgesetzt, bis auf die Frage „NAECHSTER SALDO?" eine andere als die „J"-Taste gedrückt wird. In diesem Fall wird die Endabrechnung durchgeführt, die dann die aufgelaufenen Sollzinsen, Überziehungsprovisionen, Kreditprovisionen und den neu errechneten Saldo anzeigt.

Formeln für die Verwaltung eines Girokontos

Entsprechend der üblichen Praxis wird hier mit Zinszahlen und Zins- bzw. Provisionsteilern gerechnet, die trotz einer umständlichen Definition eine vereinfachte Berechnung erlauben:

Zinszahlen (ZZ):	Saldo * Tage / 100
Zinsteiler (ZT):	360 / Zinssatz
Zinsen (Z):	Zinszahlen / Zinsteiler
Kreditprovisionsteiler (KT):	360 / Kreditprovisionssatz
Kreditprovision (KP):	Zinszahlen für Kreditlimit Zinszahlen für in Anspruch genommenen Kredit ――――― Kreditprovisionsteiler

Überz.-Provisionsteiler (UT):	360 / Überziehungsprovisionssatz
Überziehungsprovision (UP):	Zinszahlen aus Beträgen über dem Kreditlimit / Überziehungsprovisionsteiler

Beispiel

Das einfachste Beispiel für eine Zinsstaffelrechnung ist die Verwaltung Ihres eigenen Girokontos. Zur Veranschaulichung überprüfen wir die Bankauszüge aus einem zurückliegenden Quartal. Als Standardwerte setzen wir folgende Werte an:

Zinssatz:	10 %
Kreditprovisionssatz:	2 %
Überziehungsprovisionssatz:	1 %

Die in vielen Fällen zusätzlich anfallenden Kosten für Auslagen und aus Umsatzprovision lassen wir zunächst unberücksichtigt.

Da in unserem Programm keine Terminrechnung erfolgt, müssen die einzelnen Zeiten aus den Datumsangaben errechnet werden. In unserem Fall ergeben sich folgende Werte:

```
P23 – ZINSSTAFFELRECHNUNG
UEBERZ.-GRENZE          9000
ZINSSATZ                10
KREDITPROV.             2
UEBERZ. PROV.           1
SALDO                   1000
ANZAHL DER TAGE         10

FORTS., DANN X EING.    X
```

```
P23 – ZINSSTAFFELRECHNUNG
SOLLZINSEN                 5.78
UEBERZ. PROV.               .00
KREDITPROV.              333.84

NEUER SALDO             1009.62

NEUER SALDO, DANN J
```

Zur Steuerung des Programms sind folgende Hilfsgrößen bzw. Hilfsroutinen erforderlich. Zunächst werden in der Zeile 3250 die aktuellen Zinsen ermittelt. Hierbei wird das durch die INT-Funktion anfallende Abrunden durch eine vorangehende Erhöhung um 0.5 ausgeglichen. Liegt der Saldo oberhalb der Überziehungsgrenze, so wird in der Zeile 3260 die anfallende Überziehungsprovision errechnet. Als weitere Hilfsgrößen werden verwendet:

I = I + G Die Sollzinsbeträge, die für jeden Eingabezyklus anfallen, werden hier aufsummiert.

J = J + H Hier werden die fallweise anfallenden Überziehungsprovisionen aufsummiert. Da dieser Zyklus immer durchlaufen wird, muß nach jeder Aufsummierung diese Variable auf 0 zurückgesetzt werden (H = 0).

K = K + C Hier werden bei jedem Eingabezyklus die Tage aufsummiert.

ALLGEMEINE UND SPEZIELLE TABELLEN

Das Erstellen von Tabellen ist eine häufige Anwendung in BASIC. Wir haben deshalb einige einfache Grundprogramme für Sie zusammengestellt.

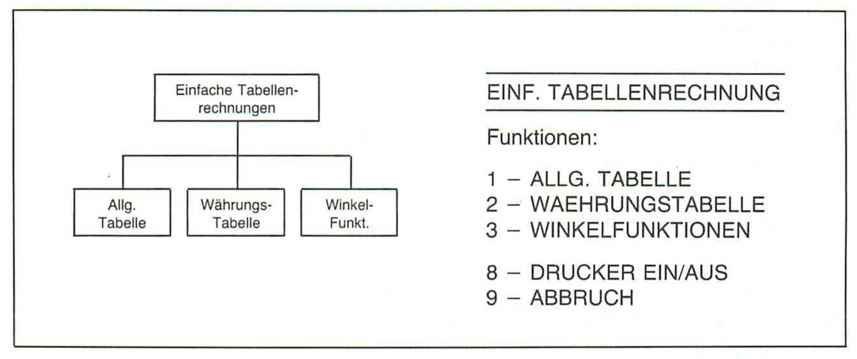


Abb. 4.21: Struktur und Menü der Tabellenrechnungsprogramme

Der Bereich von Tabellen wird vom Tabellenanfangs- und Tabellenendwert bestimmt. Diese Werte wollen wir in allen Programmen gleich, nämlich TAB1 (Tabellenanfang) und TAB2 (Tabellenende) nennen.

Für die Bestimmung der Sprünge der Tabellenwerte legen wir eine Schrittweite fest, die wir DTAB nennen.

Ein weiterer neutraler Wert ist die Länge der Tabelle je Bildschirm- bzw. Druckerseite. Diesen Wert nennen wir ZEIL. Damit können wir eine Tabelle eine Funktionsdarstellung nennen, die von TAB1 bis TAB2 mit der Schrittweite von DTAB läuft und eine dem Wert ZEIL entsprechende Anzahl von Tabellenwerten je Bildschirm- bzw. Druckerseite anzeigt.

Innerhalb einer Tabelle können nun gleichzeitig ein oder mehrere Werte je Tabelleneingangswert angezeigt werden. Begrenzt wird die Ausgabe lediglich durch die Anzahl der Spalten, die uns je Bildschirm- oder Druckerseite zur Verfügung stehen. Da der Bildschirm nur 28 Spalten zur Verfügung stellt, wollen wir uns in den Beispielen auf die Darstellung in 3 Spalten beschränken.

```
1000 ON WARNING NEXT
1010 D$="  "
1020 HL$="----------------------------"
1030 CALL CLEAR
1040 DISPLAY AT(1,1):HL$
1050 DISPLAY AT(2,1):"P3 - TABEL.RECHNUNG 12.11.83"
1060 DISPLAY AT(3,1):HL$
1070 DISPLAY AT(6,1):"1 - ALLGEMEINE TABELLE"
1080 DISPLAY AT(8,1):"2 - WAEHRUNGSTABELLE"
1090 DISPLAY AT(10,1):"3 - WINKELTABELLE"
1100 DISPLAY AT(12,1):"8 - (DRUCKER EIN/AUS) ";D$
1110 DISPLAY AT(15,1):"9 - ABBRUCH"
1120 DISPLAY AT(18,1):"FUNKTION"
1130 ACCEPT AT(18,14)VALIDATE("12389")BEEP SIZE(1):M
1140 IF M=1 THEN GOSUB 1220
1150 IF M=2 THEN GOSUB 2030
1160 IF M=3 THEN GOSUB 2480
1170 IF M=8 THEN GOSUB 2970
1180 IF M=9 THEN GOTO 1200
1190 GOTO 1030
1200 CALL CLEAR
1210 END
1220 REM ********************
1230 REM ALLGEMEINE TABELLE
1240 REM ********************
1250 GOSUB 1960
1260 REM MELDUNGSTEXT
1270 DISPLAY AT(5,1):"BEVOR SIE WEITERE EINGABEN"
1280 DISPLAY AT(6,1):"MACHEN, PRUEFEN SIE, OB SIE"
1290 DISPLAY AT(7,1):"EINE AENDERUNG DER FUNKTIO-"
1300 DISPLAY AT(8,1):"NEN IN DEN ZEILEN 1630 UND "
1310 DISPLAY AT(9,1):"1670 WUENSCHEN, FALL JA, SO"
1320 DISPLAY AT(10,1):"FUEHREN SIE DIESE ZUERST"
```

```
1330 DISPLAY AT(11,1):"AUS UND STARTEN DANACH DAS "
1340 DISPLAY AT(12,1):"PROGRAMM BITTE NEU"
1350 GOSUB 3080
1360 GOSUB 1810
1370 REM *******************
1380 REM ANZEIGEBEREICH
1390 DISPLAY AT(6,1):"BEZ. FUER VAR.    X"
1400 ACCEPT AT(6,22)SIZE(6):AB$
1410 IF LEN(AB$)=0 THEN GOTO 1400
1420 DISPLAY AT(7,1):"BEZ. FUER VAR.    Y"
1430 ACCEPT AT(7,22):BB$
1440 IF LEN(BB$)=0 THEN GOTO 1430
1450 DISPLAY AT(8,1):"BEZ. FUER VAR.    Z"
1460 ACCEPT AT(8,22):CB$
1470 IF LEN(CB$)=0 THEN GOTO 1460
1480 DISPLAY AT(10,1):"ANFANGSWERT FUER X"
1490 ACCEPT AT(10,22)VALIDATE(NUMERIC)SIZE(6):TAB1
1500 DISPLAY AT(11,1):"ENDWERT FUER X"
1510 ACCEPT AT(11,22)VALIDATE(NUMERIC)SIZE(6):TAB2
1520 DISPLAY AT(12,1):"SCHRITTWEITE FUER X"
1530 ACCEPT AT(12,22)VALIDATE(NUMERIC)SIZE(6):DTAB
1540 ZEIL=15
1550 REM *******************
1560 REM SCHLEIFE BER. UND ANZEIGE
1570 GOSUB 3080
1580 GOSUB 1810
1590 FOR X=TAB1 TO TAB2 STEP DTAB
1600 IF ZZ>ZEIL THEN GOSUB 3080 ELSE GOTO 1620
1610 GOSUB 1810
1620 GOSUB 1720
1630 DISPLAY AT(ZZ+3,1):USING "#####.##":X
1640 DISPLAY AT(ZZ+3,10):USING "#####.##":Y
1650 DISPLAY AT(ZZ+3,19):USING "#####.##":Z
1660 ZZ=ZZ+1
1670 NEXT X
1680 REM
1690 GOSUB 3020
1700 IF W$="J" THEN GOTO 1220
1710 RETURN
1720 REM *******************
1730 REM DEFINITIONEN
1740 REM *******************
1750 REM DEFINITION FUER Y
1760 Y=X*X
1770 REM DEFINITION FUER Z
1780 Z=1/X*Y
1790 RETURN
1800 REM *******************
1810 REM
1820 IF LEN(AB$)=0 THEN GOSUB 1960 ELSE 1840
1830 RETURN
1840 CALL CLEAR
1850 DISPLAY AT(1,9-LEN(AB$)):AB$
1860 DISPLAY AT(1,18-LEN(BB$)):BB$
1870 DISPLAY AT(1,27-LEN(CB$)):CB$
1880 DISPLAY AT(2,1):"----------------------------"
1890 ZZ=1
1900 RETURN
1910 RETURN
```

```
1920 GOSUB 3020
1930 IF W$="J" THEN GOTO 1000
1940 RETURN
1950 REM ********************
1960 REM KOPF 2
1970 CALL CLEAR
1980 DISPLAY AT(1,1):HL$
1990 DISPLAY AT(2,1):"P31     -     ALLGEMEINE TABELLE"
2000 DISPLAY AT(3,1):HL$
2010 RETURN
2020 REM ********************
2030 REM ********************
2040 REM WAEHRUNGSTABELLE
2050 REM BEGIN
2060 GOSUB 2410
2070 DISPLAY AT(5,1):"DOLLARKURS     DM/$"
2080 ACCEPT AT(5,22)VALIDATE(NUMERIC):DOK
2090 DISPLAY AT(6,1):"SCHILLINGKURS OS/DM"
2100 ACCEPT AT(6,22)VALIDATE(NUMERIC):OSK
2110 DISPLAY AT(8,1):"LAUFWAEHR. (DM,$,OS)"
2120 ACCEPT AT(8,22)VALIDATE(UALPHA,"DMOS$")SIZE(2):WL$
2130 IF LEN(WL$)=0 THEN GOTO 2120
2140 DISPLAY AT(10,1):"BEGINN BEI EINHEIT"
2150 ACCEPT AT(10,22)VALIDATE(NUMERIC):TAB1
2160 DISPLAY AT(11,1):"ENDE BEI EINHEIT"
2170 ACCEPT AT(11,22)VALIDATE(NUMERIC):TAB2
2180 DISPLAY AT(12,1):"SCHRITTWEITE"
2190 ACCEPT AT(12,22)VALIDATE(NUMERIC):DTAB
2200 ZEIL=15
2210 IF WL$="$" THEN DTAB=DTAB*DOK :: TAB1=TAB1*DOK ::
                                      TAB2=TAB2*DOK
2220 IF WL$="OS" THEN DTAB=DTAB*OSK/100 :: TAB1=TAB1*OSK/100 ::
                                      TAB2=TAB2*OSK
2230 ZZ=1
2240 FOR I=TAB1 TO TAB2 STEP DTAB
2250 IF ZZ<2 OR ZZ>ZEIL THEN GOSUB 2350
2260 DISPLAY AT(ZZ+4,1):USING "####.##":I
2270 DISPLAY AT(ZZ+4,11):USING "####.##":I/DOK
2280 DISPLAY AT(ZZ+4,21):USING "#####.##":I*OSK
2290 ZZ=ZZ+1
2300 NEXT I
2310 GOSUB 3020
2320 IF W$="J" THEN GOTO 2050
2330 RETURN
2340 REM ********************
2350 GOSUB 3080
2360 ZZ=1
2370 DISPLAY AT(1,1):"      DM     DOLLAR  SCHILLING"
2380 DISPLAY AT(2,1):"----------------------------"
2390 RETURN
2400 REM ********************
2410 REM KOPF 2
2420 CALL CLEAR
2430 DISPLAY AT(1,1):HL$
2440 DISPLAY AT(2,1):"P32     -     WAEHRUNGSTABELLE"
2450 DISPLAY AT(3,1):HL$
2460 RETURN
2470 REM ********************
2480 REM WINKELTABELLE
```

```
2490 REM BEGIN
2500 GOSUB 2880
2510 DISPLAY AT(8,1):"SIN/COS ODER TAN/COT"
2520 DISPLAY AT(9,1):"(S ODER T EINGEBEN"
2530 ACCEPT AT(8,22)VALIDATE("ST")SIZE(1):WF$
2540 DISPLAY AT(11,1):"BEGINN AB WINKEL"
2550 ACCEPT AT(11,22)VALIDATE(NUMERIC)SIZE(3):TAB1
2560 DISPLAY AT(12,1):"ENDE BEI WINKEL"
2570 ACCEPT AT(12,22)VALIDATE(NUMERIC)SIZE(3):TAB2
2580 DISPLAY AT(13,1):"SCHRITTWEITE"
2590 ACCEPT AT(13,22)VALIDATE(NUMERIC)SIZE(3):DTAB
2600 ZEIL=15
2610 IF WF$="S" THEN U$="WINKEL        SIN        COS"
2620 IF WF$="T" THEN U$="WINKEL        TAN        COT"
2630 GOSUB 3080
2640 GOSUB 2820
2650 FOR I=TAB1 TO TAB2 STEP DTAB
2660 IF ZZ>ZEIL THEN GOSUB 3080 ELSE GOTO 2680
2670 GOSUB 2820
2680 GOSUB 2780
2690 DISPLAY AT(ZZ+4,1):USING "###.##":I
2700 DISPLAY AT(ZZ+4,11):USING "##.####":X2
2710 DISPLAY AT(ZZ+4,21):USING "##.####":X3
2720 ZZ=ZZ+1
2730 NEXT I
2740 GOSUB 3020
2750 IF W$="J" THEN GOTO 2490
2760 RETURN
2770 REM ********************
2780 IF WF$="S" THEN X2=SIN(I*PI/180):: X3=COS(I*PI/180)
2790 IF WF$="T" THEN X2=TAN(I*PI/180):: X3=1/X2
2800 RETURN
2810 REM ********************
2820 REM
2830 CALL CLEAR
2840 DISPLAY AT(1,1):U$
2850 DISPLAY AT(2,1):HL$
2860 ZZ=1
2870 RETURN
2880 REM ********************
2890 REM KOPF WINKELTABELLE
2900 REM ********************
2910 CALL CLEAR
2920 DISPLAY AT(1,1):HL$
2930 DISPLAY AT(2,1):"P33  -  EINF. WINKELTABELLE"
2940 DISPLAY AT(3,1):HL$
2950 RETURN
2960 REM ********************
2970 REM DRUCKER EIN/AUS
2980 REM ********************
2990 IF D$=" " THEN D$="*" :: RETURN
3000 IF D$="*" THEN D$=" " :: RETURN
3010 REM ********************
3020 REM UNTERROUTINE ABBRUCH
3030 REM ********************
3040 DISPLAY AT(23,1):"WEITER, DANN J EINGEBEN"
3050 ACCEPT AT(23,25)VALIDATE("JN")SIZE(1):W$
3060 RETURN
3070 REM ********************
```

```
3080 REM UNTERROTINE FORTSETZUNG
3090 REM ********************
3100 DISPLAY AT(23,1):"FORTS., DANN X EING."
3110 ACCEPT AT(23,22)SIZE(1):N$
3120 IF N$="X" THEN CALL CLEAR :: RETURN
3130 GOTO 1000
3140 REM ********************
```

Abb. 4.22: Tabellenrechnungs-Programm

Sie erkennen am Gesamtlisting, daß auch hier von der Zeilennumerierung des Basismenüs (Abb. 4.11) abgewichen wurde.

Allgemeine Tabelle

Die allgemeine Tabelle soll abhängig von beliebig eingegebenen Funktionen Tabellenwerte errechnen. Die Laufvariable läuft dabei von einem Anfangswert TAB1 bis zu einem Endwert TAB2 in der Schrittweite DTAB so, wie Sie es wünschen. Die Überschriften für jeden zu errechnenden Tabellenwert werden dabei gemeinsam mit den abzubildenden Funktionen eingegeben.

TAB1 Tabellenanfangswert
TAB2 Tabellenendwert
X Laufvariable
DTAB Schrittweite der Laufvariablen
Y Tabellenfunktion 1
Z Tabellenfunktion 2

AB$ Überschrift Laufvariable X
BB$ Überschrift Tabellenfunktion Y
CB$ Überschrift Tabellenfunktion Z

ZEIL Anzahl der Zeilen je Bildschirm- bzw. Druckerseite (festgesetzt in Zeile 1540)

Sie erkennen im Listing des Bereichs „Allgemeine Tabelle“ (Zeilen 1220 bis 1710), daß die Definition der Tabellenfunktionen über eine einfache Zuweisung in den Zeilen 1760 (für Y) und 1780 (für Z) erfolgt. Änderungen der Funktionen sind also hier durchzuführen. Die hierzu gehörenden Druckformate finden Sie in den Zeilen 1630 bis 1650.

Beispiel

Wir können einmal eine Tabelle für die Errechnung von Quadrat- und Kubikwurzeln erstellen. Diese Tabelle soll von 3 bis 10 mit einer Schrittweite von 0,5 laufen.

Währungstabelle

Das zweite Unterprogramm soll eine Währungstabelle für die Währungen DM, Dollar und Schilling erstellen. Auch hier sollen ein Anfangs-, ein Endwert und eine Schrittweite bestimmt werden. Als Besonderheit können wir hier frei wählen, welche Währung als Laufvariable für die eingegebenen Tabellenanfangs-, -endwerte und die -schrittweite gesetzt werden soll. Diese Steuerung wird in den Zeilen 2210 und 2220 durch eine bedingte Umrechnung erreicht.

TAB1	Tabellenanfangswert
TAB2	Tabellenendwert
I	Laufvariable
DTAB	Schrittweite für I
DM	Wert für DM
$	Wert für Dollar
OS	Wert für österr. Schilling

Beispiel

Der folgende Ausdruck zeigt die erste Bildschirmseite einer Tabelle mit DM als Laufvariable und einer Schrittweite von 0,50 DM im Bereich von 1 bis 10 DM.

```
P32 – WAEHRUNGSTABELLE

DOLLARKURS DM/$                 2,65
SCHILLINGKURS DM/OS             7
LAUFWAEHRUNG (DM, $, OS)        $

BEGINN BEI EINHEIT              1
ENDE BEI EINHEIT                10
SCHRITTWEITE                    .5

FORTS., DANN X EINGEBEN         X
```

DM	DOLLAR	SCHILLING
2.65	1.00	18.55
3.98	1.50	27.83
5.30	2.00	37.10
6.63	2.50	46.38
7.95	3.00	55.65
9.28	3.50	64.93
10.60	4.00	74.20
11.93	4.50	83.48
13.25	5.00	92.75
14.58	5.50	102.03
15.90	6.00	111.30
17.23	6.50	120.58
18.55	7.00	129.85
19.88	7.50	139.13
21.20	8.00	148.40

WEITER FORTS., DANN X EING.

Winkelfunktionstabelle

Diese dritte Programmfunktion ermittelt Winkelfunktionswerte für Winkel in Altgrad. Hierbei muß in einem Vorlauf die gewünschte Winkelfunktion angegeben werden.

TAB1	Tabellenanfangswert
TAB2	Tabellenendwert
I	Laufvariable
DTAB	Schrittweite
WI	Winkel in Altgrad
WF$	Stringvariable für Tabellenauswahl
SI	Kennung für sin/cos-Tabelle
TA	Kennung für tan/cot-Tabelle

Erweiterungen

Als Erweiterungen empfehlen wir Ihnen:

- beliebiges Mischen der ausgewählten Winkelfunktionen
- beliebige Wahl eines Winkelfunktionswertes als Laufvariable

Darüber hinaus können Sie auf der Grundlage der hier gezeigten Programme spezielle Tabellen formulieren, wie zum Beispiel:

- Tabellen mit verschiedenen Rabatt- und Zuschlagssätzen
- Tabellen für technische Berechnungen
- Steuertabellen nach der allgemeinen Steuerformel

EINFACHE KALKULATIONEN

Wir stellen Ihnen hier zwei einfache Kalkulationsprobleme vor. Es sind dies einmal die Errechnung eines für eine gegebene Wohnung optimalen Stromtarifs und der Kalkulationsvergleich bei zwei unterschiedlichen Kraftfahrzeugen.

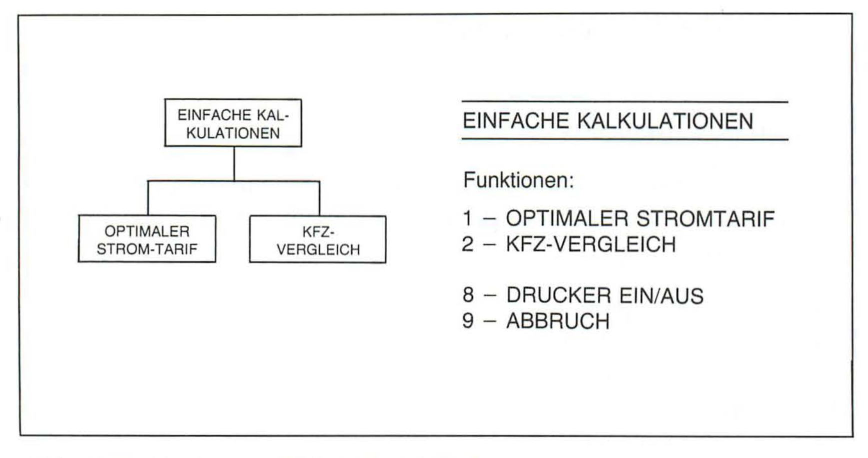


Abb. 4.23: Struktur und Menü der Kalkulationsprogramme

Die Kalkulation ist von Hause aus ein Teilgebiet der Kostenrechnung und dort häufig die Grundlage oder Kontrolle der Preisstellung von physischen Produkten und/oder Dienstleistungen. Wir benutzen sie in den Programmfunktionen 1 und 2 für einen Kostenvergleich. Voraussetzung hierfür ist dabei jeweils die Grundlage vergleichbarer Leistungen, was uns zumindest bei den ausgewählten Problemen keine großen Schwierigkeiten bereitet.

```
1000 ON WARNING NEXT
1010 D$=" "
1020 HL$="----------------------------"
1030 REM HAUPTMENU
1040 CALL CLEAR
1050 DISPLAY AT(1,1):HL$
1060 DISPLAY AT(2,1):"P4 -  EINFACHE KALKULATIONEN"
1070 DISPLAY AT(3,1):HL$
1080 DISPLAY AT(6,1):"1 - OPTIMALER STROMTARIF"
1090 DISPLAY AT(8,1):"2 - KFZ-VERGLEICH"
1100 DISPLAY AT(12,1):"8 - DRUCKER EIN/AUS ";D$
1110 DISPLAY AT(15,1):"9 - ABBRUCH"
1120 DISPLAY AT(18,1):"FUNKTION"
1130 ACCEPT AT(18,14)VALIDATE("1289")BEEP SIZE(1):M
1140 IF M=1 THEN GOSUB 1210
1150 IF M=2 THEN GOSUB 1780
1160 IF M=8 THEN GOSUB 2620
1170 IF M=9 THEN GOTO 1190
1180 GOTO 1030
1190 CALL CLEAR
1200 END
1210 REM ********************
1220 REM OPTIMALER STROMTARIF
1230 REM ********************
1240 GOSUB 1700
1250 REM DEFINITIONEN
1260 SK1=0.179 :: SK2=0.149
1270 VK11=6.6 :: VK12=1.75
1280 VK21=12.9 :: VK22=3.40
1290 GI=2
1300 REM ********************
1310 REM DATENERFASSUNG
1320 REM ********************
1330 DISPLAY AT(5,1):"ANZAHL DER RAEUME"
1340 ACCEPT AT(5,22)VALIDATE(DIGIT)SIZE(1):A
1350 DISPLAY AT(6,1):"VORAUSSICHTLICHER"
1360 DISPLAY AT(7,1):"MONATSVERBRAUCH"
1370 DISPLAY AT(8,1):"IN KILOWATTSTUNDEN"
1380 ACCEPT AT(6,22)VALIDATE(NUMERIC)SIZE(7):B
1390 REM ********************
1400 REM BERECHNUNG
1410 IF A<GI THEN GOSUB 1560 ELSE GOSUB 1610
1420 GOSUB 1560
1430 GOSUB 1610
1440 IF P1<P2 THEN OT$="I" ELSE OT$="II"
1450 DISPLAY AT(10,1):"DER OPTIMALE STROMTARIF"
1460 DISPLAY AT(11,1):"IST DER TARIF"
1470 DISPLAY AT(11,15):OT$
1480 DISPLAY AT(13,1):"KOSTEN BEI TARIF I"
1490 DISPLAY AT(13,20):USING ("#####.##"):P1
1500 DISPLAY AT(14,1):"KOSTEN BEI TARIF II"
1510 DISPLAY AT(14,20):USING ("#####.##"):P2
1520 GOSUB 2680
1530 IF W$="J" THEN GOTO 1210
1540 RETURN
1550 REM ********************
1560 REM TARIF I
1570 REM ********************
1580 P1=A*VK11+B*SK1
```

```
1590 P2=A*VK21+B*SK2
1600 RETURN
1610 REM TARIF II
1620 P1=VK11+(A-GI)*VK12+B*SK1
1630 P2=VK21+(A-GI)*VK22+B*SK2
1640 RETURN
1650 REM *******************
1660 GOSUB 2680
1670 IF W$="J" THEN GOTO 1210
1680 RETURN
1690 REM *******************
1700 REM UEBERSCHRIFT STROMTARIF
1710 REM *******************
1720 CALL CLEAR
1730 DISPLAY AT(1,1):HL$
1740 DISPLAY AT(2,1):"P41  -  OPTIMALER STROMTARIF"
1750 DISPLAY AT(3,1):HL$
1760 RETURN
1770 REM *******************
1780 REM KFZ-VERGLEICH
1790 REM *******************
1800 K1=0 :: K2=0
1810 GOSUB 2540
1820 REM
1830 DISPLAY AT(5,1):"DIESES PROGRAMM BERECHN. DIE"
1840 DISPLAY AT(6,1):"GESAMTKOSTEN, DIE ALTERNATIV"
1850 DISPLAY AT(7,1):"BEI VERSCHIEDENEN KRAFTFAHR-"
1860 DISPLAY AT(8,1):"ZEUGEN ANFALLEN"
1870 GOSUB 2740
1880 CALL CLEAR
1890 GOSUB 2540
1900 DISPLAY AT(5,1):"ART          KFZ-1    KFZ-2"
1910 DISPLAY AT(6,1):"----------------------------"
1920 REM *******************
1930 REM DATENERFASSUNG
1940 REM *******************
1950 DISPLAY AT(8,1):"ANSCHAFF."
1960 ACCEPT AT(8,13)VALIDATE(NUMERIC)SIZE(7):A1
1970 ACCEPT AT(8,22)VALIDATE(NUMERIC)SIZE(7):A2
1980 DISPLAY AT(9,1):"DM/LITER"
1990 ACCEPT AT(9,13)VALIDATE(NUMERIC)SIZE(4):B1
2000 IF B1>2 THEN GOTO 1990
2010 ACCEPT AT(9,22)VALIDATE(NUMERIC)SIZE(4):B2
2020 IF B2>2 THEN GOTO 2010
2030 DISPLAY AT(10,1):"L/100 KM"
2040 ACCEPT AT(10,13)VALIDATE(NUMERIC)SIZE(5):C1
2050 IF C1>25 THEN GOTO 2040
2060 ACCEPT AT(10,22)VALIDATE(NUMERIC)SIZE(5):C2
2070 IF C2>25 THEN GOTO 2060
2080 DISPLAY AT(11,1):"VERS./JAHR"
2090 ACCEPT AT(11,13)VALIDATE(NUMERIC)SIZE(6):D1
2100 IF D1>5000 THEN GOTO 2090
2110 ACCEPT AT(11,22)VALIDATE(NUMERIC)SIZE(6):D2
2120 IF D2>5000 THEN GOTO 2130
2130 DISPLAY AT(12,1):"STEUER/JAHR"
2140 ACCEPT AT(12,13)VALIDATE(NUMERIC)SIZE(6):E1
2150 IF E1>2000 THEN GOTO 2140
2160 ACCEPT AT(12,22)VALIDATE(NUMERIC)SIZE(6):E2
2170 IF E2>2000 THEN GOTO 2180
```

```
2180 DISPLAY AT(13,1):"KM/JAHR"
2190 ACCEPT AT(13,13)VALIDATE(DIGIT)SIZE(5):G
2200 DISPLAY AT(13,22):USING ("#####"):G
2210 DISPLAY AT(14,1):"ZEITRAUM"
2220 ACCEPT AT(14,13)VALIDATE(NUMERIC)SIZE(3):H
2230 IF H>10 THEN GOTO 2220
2240 DISPLAY AT(14,22):USING ("###"):H
2250 DISPLAY AT(15,1):"RESTWERT"
2260 ACCEPT AT(15,13)VALIDATE(NUMERIC)SIZE(7):F1
2270 ACCEPT AT(15,22)VALIDATE(NUMERIC)SIZE(7):F2
2280 GOSUB 2740
2290 CALL CLEAR
2300 DISPLAY AT(1,1):"----------------------------"
2310 DISPLAY AT(2,1):"JAHR  G     KFZ-1      KFZ-2"
2320 DISPLAY AT(3,1):"----------------------------"
2330 FOR I=1 TO H
2340 K1=K1+(A1-F1)/H+(D1+E1+B1+C1*G/100)
2350 K2=K2+(A2-F2)/H+(D2+E2+B2+C2*G/100)
2360 IF K1<K2 THEN K$="1" ELSE K$="2"
2370 GOSUB 2460
2380 NEXT I
2390 DISPLAY AT(17,1):"UNTER EINBEZIEHUNG ALLER"
2400 DISPLAY AT(18,1):"KOSTEN IST DAS GUENSTIGSTE"
2410 DISPLAY AT(19,1):"FAHRZEUG DAS KFZ"
2420 DISPLAY AT(19,18):K$
2430 GOSUB 2740
2440 GOTO 1780
2450 REM ********************
2460 REM ANZEIGEROUTINE
2470 REM ********************
2480 DISPLAY AT(I+4,1):USING ("##"):I
2490 DISPLAY AT(I+4,7):K$
2500 DISPLAY AT(I+4,10):USING ("######.##"):K1
2510 DISPLAY AT(I+4,20):USING ("######.##"):K2
2520 RETURN
2530 REM ********************
2540 REM UEBERSCHRIFT KFZ-VERGLEICH
2550 REM ********************
2560 CALL CLEAR
2570 DISPLAY AT(1,1):HL$
2580 DISPLAY AT(2,1):"P42    -    KFZ-VERGLEICH"
2590 DISPLAY AT(3,1):HL$
2600 RETURN
2610 REM ********************
2620 REM ********************
2630 REM DRUCKER EIN/AUS
2640 REM ********************
2650 IF D$=" " THEN D$="*" :: RETURN
2660 IF D$="*" THEN D$=" " :: RETURN
2670 REM ********************
2680 REM ABBRUCH
2690 REM ********************
2700 DISPLAY AT(23,1):"WEITER, DANN J EINGEBEN"
2710 ACCEPT AT(23,25)VALIDATE("JN")SIZE(1):W$
2720 RETURN
2730 REM ********************
2740 REM STOP/WEITER
2750 REM ********************
2760 DISPLAY AT(23,1):"FORTS., DANN X EING."
```

```
2770 ACCEPT AT(23,22)SIZE(1):N$
2780 IF N$="X" THEN RETURN
2790 GOTO 1000
2800 REM ********************
```

Abb. 4.24: Kalkulations-Programm

Optimaler Stromtarif

Bei Bezug einer neuen Wohnung oder bei einer Veränderung der Stromverbrauchsgewohnheiten empfiehlt sich häufig die Errechnung der Stromkosten bei alternativen Tarifen. Je nach geographischer Zugehörigkeit müssen hierzu unterschiedliche Ansätze herangezogen werden. Unserem Beispiel (Passau) liegen für den Privathaushalt zwei Tarife (Tarif I und Tarif II) zugrunde, denen folgende Kosten zugeordnet sind:

Tarif I

- VK11 = 6,60 DM für die ersten beiden Räume
- VK12 = 1,75 DM für jeden weiteren Raum
- SK1 = 17,90 Pf für jede Kilowattstunde Verbrauch

Tarif II

- VK21 = 12,90 DM für die ersten beiden Räume
- VK22 = 3,40 DM für jeden weiteren Raum
- SK2 = 14,90 Pf für jede Kilowattstunde Verbrauch

Die Festlegungen erfolgen im Bereich Definitionen.

```
1250 REM DEFINITIONEN
1260 SK1 = 0,179 : : SK2 = 0,149
1270 VK11 = 6,6 : : VK12 = 1,75
1280 VK21 = 12,9 : : VK22 = 3,40
1290 GI = 2
```

Die Konstante GI wird als Schalter für die Anzahl der Räume festgelegt, die mit dem untersten Pauschalsatz VK11 bzw. VK21 belegt sind. In unserem Fall sind dies zwei Räume, also GI = 2.

Errechnet werden die für einen veranschlagten Kilowattverbrauch (geplant bei Neubezug einer Wohnung, Ist bei einer Nachrechnung) entstehenden Kosten mit der Routine

```
1560 REM TARIF I
1570 REM ******************
1580 P1 = A*VK11 + B*SK1
1590 P2 = A*VK21 + B*SK2
1600 RETURN
```

bei ein bis zwei Räumen und mit der Routine

```
1610 REM TARIF II
1620 P1 = VK11 + (A - GI)*VK12 + B*SK1
1630 P2 = VK21 + (A - GI)*VK22 + B*SK2
1640 RETURN
```

bei drei oder mehr Räumen.

Beispiel

Nach Zukauf weiterer Stromverbraucher möchte eine Familie, die in einer Dreizimmerwohnung wohnt, den für sie optimalen Stromtarif errechnen. Die letzte Abrechnung weist einen Monatsverbrauch von 440 kWh aus, und es wird mit einem Anstieg des Verbrauchs um 10 % gerechnet.

```
P41 - OPTIMALER STROMTARIF

ANZAHL DER RAEUME              3
VORAUSSICHTLICHER              484
MONATSVERBRAUCH
IN KILOWATTSTUNDEN

DER OPTIMALE STROMTARIF
IST DER TARIF II

KOSTEN BEI TARIF I              94,99
KOSTEN BEI TARIF II             88,42

WEITER, DANN J EINGEBEN
```

Erweiterungen

Für den Ausbau dieses einfachen Programms sind Erweiterungen denkbar wie:

- die Anzeige des Verbrauchs, bei dem sich die Kosten beider Tarife überschneiden, oder
- die Anzeige von Verbrauchsmengen und Kosten in einem Verbrauchsbereich, der 20 % oberhalb und unterhalb des eingesetzten Verbrauchs liegt.

KFZ-Vergleich

Mit dieser Programmfunktion greifen wir ein Problem auf, das beim Kauf eines PKWs entsteht und außerdem zu den liebsten Freizeitbeschäftigungen vieler Mitbürger zählt, nämlich die Fragestellung:

- Welches Auto kaufe ich mir, oder
- wie kann ich trotz eines hohen Anschaffungspreises die Kaufentscheidung für ein teures Fahrzeug sachlich begründen?

Datenerfassung

Wir erfassen für jedes Fahrzeug

A1 und A2 = Anschaffungskosten
B1 und B2 = Kraftstoffpreis (Normal, Super, Diesel)
C1 und C2 = Kraftstoffverbrauch auf 100 km
D1 und D2 = Jahresversicherungskosten
E1 und E2 = Jahressteuer
F1 und F2 = Restwert (Wiederverkaufs- bzw. Schrottwert)

und für beide Fahrzeuge gemeinsam

G = geplante Kilometerleistung im Jahr
H = Betrachtungszeitraum bis zum Verkauf oder zur Verschrottung

Damit keine unsinnigen Zahlen in die Rechnung eingehen, haben wir die Eingabebereiche auf plausible Größen eingegrenzt.

Danach ist unsere Rechnung sehr einfach. Zunächst ermitteln wir den Kostenblock, der sich aus der Beschaffung und späteren Veräußerung des Fahrzeugs errechnet, und verteilen diesen gleichmäßig auf die

```
1920 REM ********************
1930 REM DATENERFASSUNG
1940 REM ********************
1950 DISPLAY AT(8,1):"ANSCHAFF."
1960 ACCEPT AT(8,13)VALIDATE(NUMERIC)SIZE(7):A1
1970 ACCEPT AT(8,22)VALIDATE(NUMERIC)SIZE(7):A2
1980 DISPLAY AT(9,1):"DM/LITER"
1990 ACCEPT AT(9,13)VALIDATE(NUMERIC)SIZE(4):B1
2000 IF B1>2 THEN GOTO 1990
2010 ACCEPT AT(9,22)VALIDATE(NUMERIC)SIZE(4):B2
2020 IF B2>2 THEN GOTO 2010
2030 DISPLAY AT(10,1):"L/100 KM"
2040 ACCEPT AT(10,13)VALIDATE(NUMERIC)SIZE(5):C1
2050 IF C1>25 THEN GOTO 2040
2060 ACCEPT AT(10,22)VALIDATE(NUMERIC)SIZE(5):C2
2070 IF C2>25 THEN GOTO 2060
2080 DISPLAY AT(11,1):"VERS./JAHR"
2090 ACCEPT AT(11,13)VALIDATE(NUMERIC)SIZE(6):D1
2100 IF D1>5000 THEN GOTO 2090
2110 ACCEPT AT(11,22)VALIDATE(NUMERIC)SIZE(6):D2
2120 IF D2>5000 THEN GOTO 2130
2130 DISPLAY AT(12,1):"STEUER/JAHR"
2140 ACCEPT AT(12,13)VALIDATE(NUMERIC)SIZE(6):E1
2150 IF E1>2000 THEN GOTO 2140
2160 ACCEPT AT(12,22)VALIDATE(NUMERIC)SIZE(6):E2
2170 IF E2>2000 THEN GOTO 2180
2180 DISPLAY AT(13,1):"KM/JAHR"
2190 ACCEPT AT(13,13)VALIDATE(DIGIT)SIZE(5):G
2200 DISPLAY AT(13,22):USING ("#####"):G
2210 DISPLAY AT(14,1):"ZEITRAUM"
2220 ACCEPT AT(14,13)VALIDATE(NUMERIC)SIZE(3):H
2230 IF H>10 THEN GOTO 2220
2240 DISPLAY AT(14,22):USING ("###"):H
2250 DISPLAY AT(15,1):"RESTWERT"
2260 ACCEPT AT(15,13)VALIDATE(NUMERIC)SIZE(7):F1
2270 ACCEPT AT(15,22)VALIDATE(NUMERIC)SIZE(7):F2
```

Abb. 4.25: Programmteil Datenerfassung

Nutzungszeit. Entsprechend den anderen variablen Kosten ermitteln wir dann für jedes Jahr die Jahreskosten und zeigen diese in einer Tabelle an. Gleichzeitig wird in einer speziellen Spalte G ausgegeben, welches Fahrzeug das günstigste ist. Die Berechnung und Anzeige erfolgt dabei in einer Schleife:

```
2330 FOR I = 1 TO H
2340 K1 = K1 + (A1 − F1)/H + (D1 + E1 + B1 + C1*G/100)
2350 K2 = K2 + (A2 − F2)/H + (D2 + E2 + B2 + C2*G/100)
```

Obwohl der Zusammenhang sehr einfach ist, haben wir eine tabellarische Darstellung gewählt, um so den Ausgabenstrom für das private Haushaltsbudget aufzuzeigen.

Beispiel

Im folgenden Beispiel stehen wir vor folgender Fragestellung: Wir haben die Gelegenheit, ein sehr gut erhaltenes, repräsentatives Gebrauchtfahrzeug (Baujahr 1976) zu einem Preis von 8.450,– DM zu kaufen. Auf der anderen Seite gefällt uns aber das neueste Modell eines Mittelklassewagens zum Preis von 14.450,– DM. Aus den Angaben der Verkäufer kennen wir die einzelnen Kraftstoffverbräuche (13.5 l für Kfz 1 und 8 l für Kfz 2), die wir entsprechend ansetzen. Für das neue Fahrzeug wollen wir eine Vollkaskoversicherung abschließen. Beide Fahrzeuge würden wir etwa fünf Jahre mit einer Jahreskilometerleistung von 25.000 km fahren. Für das Fahrzeug 1 rechnen wir nach Ablauf der fünf Jahre mit einem Schrottwert von 350 DM, bei Fahrzeug 2 hoffen wir auf einen Wiederverkaufswert von 5.500,– DM.

P42 – KFZ-VERGLEICH

ART	KFZ-1	KFZ-2
ANSCHAFF.	8450	14450
DM/LITER	1.4	1.4
L/100 KM	13.5	8
VERS./JAHR	850	1750
STEUER/JAHR	273	242
KM/JAHR	25000	25000
ZEITRAUM	5	5
RESTWERT	350	5500

FORTS., DANN X EINGEBEN X

JAHR	G	KFZ-1	KFZ-2
1	2	6119,40	5783,40
2	2	12238,80	11566,80
3	2	18358,20	17350,20
4	2	24477,60	23133,60
5	2	30597,00	28917,00

UNTER EINBEZIEHUNG ALLER
KOSTEN IST DAS GUENSTIGSTE
FAHRZEUG DAS KFZ 2

FORTS., DANN X EINGEBEN

Wenn Sie das Beispiel mit verschiedenen Zeiträumen, km-Leistungen und Restwerten ansetzen, erkennen Sie, wie sensibel dieses einfache Modell ist und wie leicht Sie es bei entsprechenden Größen so beeinflussen können, daß die von Ihnen gefällte Entscheidung sachlich begründet werden kann. Wir wünschen Ihnen hierbei viel Vergnügen.

Erweiterungen

Für eine Verbesserung dieser einfachen Rechnung bietet sich an:

- Einbau von notwendigen Zusatzausgaben für Reparaturen, die zeitneutral oder zu bestimmten Zeitpunkten anfallen.
- Berücksichtigung eventueller Kreditkosten, die für die Beschaffung des teureren Fahrzeugs anfallen, sowie die
- Berücksichtigung spezieller Kosten (z.B. erhöhter Ölverbrauch bei einem älteren Fahrzeug).

EINFACHE INVESTITIONSRECHNUNGEN

In diesem Abschnitt wollen wir drei einfache Rechnungen durchführen, die wir unter dem Oberbegriff Investitionsrechnungen zusammengefaßt haben. Unter einer Investition verstehen wir allgemein die langfristige Anlage von Kapital zum Zwecke der Veränderung der Produktion. Wir haben hierzu drei einfache Rechnungen ausgewählt, die wir Ihnen zur Anwendung empfehlen.

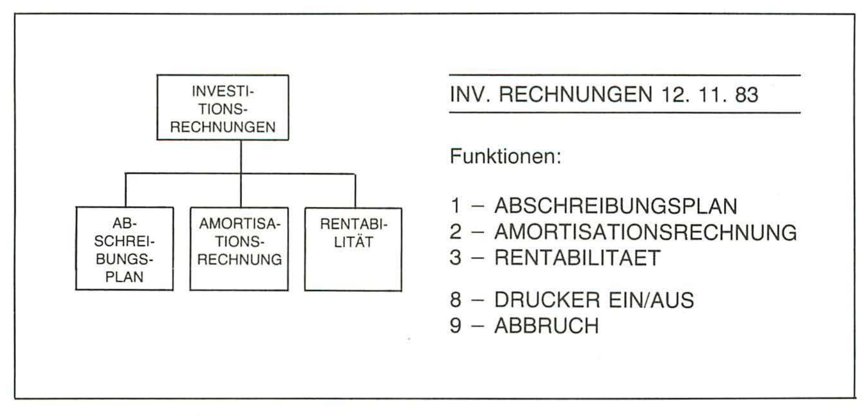


Abb. 4.26: Struktur und Menü der Investitionsrechnungs-Programme

```
1000 ON WARNING NEXT
1010 D$=" "
1020 HL$="----------------------------"
1030 CALL CLEAR
1040 DISPLAY AT(1,1):HL$
1050 DISPLAY AT(2,1):"P5 - INV.RECHNUNGEN 12.11.83"
1060 DISPLAY AT(3,1):HL$
1070 DISPLAY AT(6,1):"1 - ABSCHREIBUNGSPLAN"
1080 DISPLAY AT(8,1):"2 - AMORTISATIONSRECHNUNG"
1090 DISPLAY AT(10,1):"3 - RENTABILITAET"
1100 DISPLAY AT(12,1):"8 - DRUCKER EIN/AUS ";D$
1110 DISPLAY AT(15,1):"9 - ABBRUCH"
1120 DISPLAY AT(18,1):"FUNKTION"
1130 ACCEPT AT(18,14)VALIDATE("12389")BEEP SIZE(1):M
1140 IF M=1 THEN GOSUB 1230
1150 IF M=2 THEN GOSUB 1780
1160 IF M=3 THEN GOSUB 2120
1170 IF M=8 THEN GOSUB 2450
1180 IF M=9 THEN GOTO 1200
1190 GOTO 1030
1200 CALL CLEAR
1210 END
1220 REM ********************
1230 REM ********************
1240 REM ABSCHREIBUNGSPLAN
1250 REM BEGIN
1260 X=0 :: E=0 :: Z=0
1270 GOSUB 1720
1280 REM
1290 DISPLAY AT(5,1):"ANSCHAFFUNGSWERT"
1300 ACCEPT AT(5,22)VALIDATE(DIGIT)SIZE(7):A
1310 IF A>99999 THEN GOTO 1300
1320 DISPLAY AT(6,1):"NUTZUNGSDAUER"
1330 ACCEPT AT(6,22)VALIDATE(DIGIT)SIZE(2):B
1340 IF B>12 THEN GOTO 1330
1350 DISPLAY AT(7,1):"AFA-SATZ FUER DEGR."
1360 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(2):C
1370 IF C>35 THEN 1360
1380 REM ********************
1390 REM AUSWERTUNGSBEREICH
1400 X=0 :: E=0 :: CALL CLEAR
1410 X=X+1
1420 IF A*C/100<A/B THEN GOSUB 1500 ELSE D=A*C/100
1430 REM
1440 REM
1450 A=A-D
1460 B=B-1
1470 Z=Z+1
1480 IF A=0 THEN E=E+D :: GOTO 1630 ELSE GOSUB 1550
1490 GOTO 1410
1500 REM
1510 D=A/B
1520 DISPLAY AT(Z+3,27):USING "###":"L"
1530 RETURN
1540 REM ********************
1550 REM
1560 DISPLAY AT(1,1):"JAHR    AFA-WERT  BUCHWERT"
1570 DISPLAY AT(2,1):"----------------------------"
1580 DISPLAY AT(Z+3,2):USING "##":X
```

```
1590 DISPLAY AT(Z+3,8):USING "######.##":D
1600 DISPLAY AT(Z+3,18):USING "######.##":A
1610 E=E+D
1620 RETURN
1630 REM
1640 DISPLAY AT(18,1):"SUMME AFA GESAMT"
1650 DISPLAY AT(18,18):USING "######.##":E
1660 A=1 :: D=D-1
1670 GOSUB 1550
1680 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
1690 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
1700 IF W$="J" THEN GOTO 1250
1710 RETURN
1720 REM KOPF 2
1730 CALL CLEAR
1740 DISPLAY AT(1,1):HL$
1750 DISPLAY AT(2,1):"P51 - DEGR. ABSCHREIBUNGSPL."
1760 DISPLAY AT(3,1):HL$
1770 RETURN
1780 REM KAPITALRUECKFLUSSZEIT
1790 REM BEGIN
1800 GOSUB 2050
1810 REM
1820 DISPLAY AT(5,1):"KAPITALEINSATZ"
1830 ACCEPT AT(5,22)VALIDATE(NUMERIC)SIZE(7):A
1840 DISPLAY AT(6,1):"JAEHRL. GEWINN"
1850 ACCEPT AT(6,22)VALIDATE(NUMERIC)SIZE(7):B
1860 DISPLAY AT(7,1):"ABSCHR (%)"
1870 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(3):D
1880 REM RECHENROUTINE
1890 E=A/(B+A*D/100)
1900 F=INT(E)
1910 G=INT((E-INT(E))*12)
1920 H=INT(((E-INT(E))*12-G)*30)
1930 DISPLAY AT(10,1):"HIERAUS ERGIBT SICH EINE"
1940 DISPLAY AT(11,1):"AMORTISATION NACH"
1950 DISPLAY AT(13,1):USING "###":F
1960 DISPLAY AT(13,5):"JAHREN"
1970 DISPLAY AT(14,1):USING "###":G
1980 DISPLAY AT(14,5):"MONATEN UND"
1990 DISPLAY AT(15,1):USING "###":H
2000 DISPLAY AT(15,5):"TAGEN"
2010 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
2020 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
2030 IF W$="J" THEN GOTO 1790
2040 RETURN
2050 REM KOPF 2
2060 CALL CLEAR
2070 DISPLAY AT(1,1):HL$
2080 DISPLAY AT(2,1):"P52 -  AMORTISATIONSRECHNUNG"
2090 DISPLAY AT(3,1):HL$
2100 RETURN
2110 REM ********************
2120 REM RENTABILITAET
2130 REM BEGIN
2140 GOSUB 2380
2150 REM
2160 DISPLAY AT(5,1):"KAPITALEINSATZ"
2170 ACCEPT AT(5,22)VALIDATE(NUMERIC)SIZE(7):A
```

```
2180 DISPLAY AT(6,1):"JAEHRL. GEWINN"
2190 ACCEPT AT(6,22)VALIDATE(NUMERIC)SIZE(7):B
2200 DISPLAY AT(7,1):"JAEHRL. KOSTENERSP."
2210 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(7):C
2220 R=B*100/A
2230 ROI=C*100/A
2240 REM ANZEIGEROUTINE
2250 DISPLAY AT(10,1):"HIERAUS ERGIBT SICH EINE"
2260 DISPLAY AT(11,1):"RENTABILITAET VON"
2270 DISPLAY AT(13,1):USING "##.##":R
2280 DISPLAY AT(13,8):"%"
2290 DISPLAY AT(15,1):"UND EIN"
2300 DISPLAY AT(16,1):"RETURN OF INVESTMENT VON"
2310 DISPLAY AT(18,1):USING "##.##":ROI
2320 DISPLAY AT(18,8):"%"
2330 DISPLAY AT(22,1):"WEITER, DANN J EINGEBEN"
2340 ACCEPT AT(22,25)VALIDATE("JN")SIZE(1):W$
2350 IF W$="J" THEN GOTO 1790
2360 RETURN
2370 REM ********************
2380 REM KOPF 3
2390 CALL CLEAR
2400 DISPLAY AT(1,1):HL$
2410 DISPLAY AT(2,1):"P53    -    RENTABILITAET"
2420 DISPLAY AT(3,1):HL$
2430 RETURN
2440 REM ********************
2450 REM DRUCKER EIN/AUS
2460 IF D$=" " THEN D$="*" :: RETURN
2470 IF D$="*" THEN D$=" " :: RETURN
2480 REM ********************
```

Abb. 4.27: Investitionsrechnungs-Programm

Abschreibungsplan

Die Abschreibung wird im Rechnungswesen eines Betriebs benutzt, um in der Finanzbuchhaltung eine periodegerechte Verteilung der Ausgaben und das Entsprechende in der Kostenrechnung für die Kosten vornehmen zu können. Hierbei werden die Anschaffungskosten über die gesamte Nutzungsdauer nach einem der dafür üblichen Verfahren verteilt. Üblich sind vor allem folgende Verfahren:

Abschreibung mit regelmäßigen Quoten

1. bei konstanten Quoten die

- lineare Abschreibung, bei der der Anschaffungswert über die gesamte Nutzungsdauer in konstanten Beträgen verteilt wird, und die

- Zinseszinsabschreibung, bei der angenommen wird, daß die Abschreibungsbeträge Zinsen bringen und dies bei der Festsetzung der linearen Beträge berücksichtigt wird,

2. bei fallenden Quoten die

- geometrisch degressive Abschreibung, bei der mit einem konstanten Abschreibungssatz vom Restbuchwert abgeschrieben wird, und die
- arithmetisch degressive Abschreibung, bei der die Restnutzungsdauer in die Berechnung der degressiven Sätze eingeht,

3. bei steigenden Quoten die

- progressive Abschreibung, die immer dann angesetzt werden könnte (sofern zulässig), wenn ein steigender Umsatz mit dem Investitionsobjekt verbunden ist.

Abschreibung mit unregelmäßigen Quoten

- nach dem Rohgewinn und
- nach der Maßgabe der tatsächlichen Beanspruchung.

In unserem Beispiel haben wir die degressive Abschreibung gewählt. Hierbei lassen wir zu, daß im Verlauf der Abschreibung ein Übergang zur linearen Abschreibung möglich wird. Weiterhin wollen wir nach Ablauf der Nutzungszeit einen Restwert von 1 DM realisieren.

Datenerfassung

In der Datenerfassung ordnen wir folgende Daten zu:

A = Anschaffungswert des Investitionsobjektes
B = Nutzungsdauer, über die die Abschreibung des Inv.-Objektes verteilt werden soll
C = AfA-Satz (Abschreibungssatz für Anlagen) in %

Wir begrenzen den AfA-Satz dabei auf 35 %. Sollten Sie höhere AfA-Sätze zulassen, so müssen Sie die Zeile 1370 ändern.

```
1290 DISPLAY AT(5,1):"ANSCHAFFUNGSWERT"
1300 ACCEPT AT(5,22)VALIDATE(DIGIT)SIZE(7):A
1310 IF A>99999 THEN GOTO 1300
1320 DISPLAY AT(6,1):"NUTZUNGSDAUER"
1330 ACCEPT AT(6,22)VALIDATE(DIGIT)SIZE(2):B
1340 IF B>12 THEN GOTO 1330
1350 DISPLAY AT(7,1):"AFA-SATZ FUER DEGR."
1360 ACCEPT AT(7,22)VALIDATE(NUMERIC)SIZE(2):C
1370 IF C>35 THEN 1360
```

Abb. 4.28: Programmteil Datenerfassung

Berechnung

Wir gehen bei der Berechnung zunächst von der degressiven Abschreibung aus und überprüfen jeweils vor jedem Rechenzyklus, ob ein Übergang auf die lineare Abschreibung möglich ist. Dies ist dann der Fall, wenn der fiktive lineare Abschreibungsbetrag für die Restnutzungsdauer höher als der degressive Abschreibungsbetrag ist. Die Prüfung und Berechnung des Abschreibungssatzes wird in Zeile 1420 ausgeführt bzw. veranlaßt:

```
1420 IF A*C/100 < A/B THEN GOSUB 1500 ELSE D = A*C/100
```

Anzeige

Angezeigt wird für jedes Jahr jeweils der Buchwert, der Abschreibungsbetrag und ein Kennzeichen, das anzeigt, wenn der Übergang zur linearen Abschreibung erfolgt ist.

Beispiel

In unserem Beispiel wollen wir den Abschreibungsplan für ein Kraftfahrzeug ermitteln, das wir zu einem Preis von 18.000 DM beschaffen und für das wir eine Nutzungsdauer von 5 Jahren ansetzen. Als Abschreibungssatz wählen wir einmal 25 % und prüfen den Verlauf der Beträge. Sollten Sie die Möglichkeit der Abschreibung Ihres Fahrzeuges im Rahmen Ihrer Berufstätigkeit besitzen, so wird Ihnen diese Methode sicher bekannt sein.

P51 – DEGR. ABSCHREIBUNGSPL.

ANSCHAFFUNGSWERT	18.000
NUTZUNGSDAUER	5
AFA-SATZ	25 %

JAHR	AFA-WERT	BUCHWERT L
1	4500,00	13500,00
2	3375,00	10125,00 L
3	3375,00	6775,00 L
4	3375,00	3375,00 L
5	3374,00	1,00 L

SUMME AFA GESAMT 18,000

WEITER, DANN J EINGEBEN

Amortisationsrechnung

Im Rahmen der Amortisationsrechnung wird die Dauer der Amortisation (Verflüssigung) des investierten Kapitals errechnet. Vom Ansatz her sehr einfach, kann mit dieser Methode eine erste „Daumenrechnung" für die Ermittlung der Zeit durchgeführt werden, bis zu der investiertes Kapital wieder vollständig „zurückgeflossen" ist.

Werden die Zinsen nicht berücksichtigt, so werden hauptsächlich drei Ansätze unterschieden:

1. Amortisation für eine Gründungsinvestition

$$\frac{\text{Kapitaleinsatz}}{\text{jährlicher Rückfluß (cash flow)}}$$

2. Amortisation einer Erweiterungsinvestition

$$\frac{\text{Kapitaleinsatz}}{\text{zusätzlicher Jahresgewinn}}$$

3. Amortisation einer Rationalisierungsinvestition

$$\frac{\text{Kapitaleinsatz}}{\text{jährliche Kostenersparnis}}$$

Wir haben für unser Programm Ansatz 2 gewählt. Sie erkennen jedoch, daß Sie auch die anderen Ansätze damit leicht erfassen. Sie müssen nur die Abfrage nach dem Jahresgewinn entsprechend interpretieren.

Beispiel

Wir haben eine private Hühnerfarm mit 5 Hühnern und wollen im Rahmen einer Erweiterungsinvestition zwei Hühner zukaufen. Für den Fütterungsvorgang und die Unterbringung sind keine besonderen Aufwendungen erforderlich, da die Hühner von den Küchenabfällen unserer Familie sehr gut leben können. Das bedeutet, daß wir ausschließlich von einer Gewinnerhöhung ausgehen können. Eine Gewinnsteuer oder sonstige Steuer fällt ebenfalls nicht an, da es sich um sehr geringe Beträge handelt und die Hühner außerdem auf den Namen der Ehefrau geschrieben sind, die sonst keinem Erwerb nachgeht. Wir rechnen mit zusätzlichen 650 Eiern, die wir für insgesamt 65,– DM verkaufen können.

```
P52 – AMORTISATIONSRECHNUNG
KAPITALEINSATZ            34,80
JAEHRL. GEWINN            65,00
ABSCHREIBUNG (%)           0

HIERAUS ERGIBT SICH EINE
AMORTISATION NACH

 0 JAHREN
 7 MONATEN UND
17 TAGEN

WEITER, DANN J EINGEBEN
```

Erweiterung

Versuchen Sie, den einfachen Ansatz um zusätzliche Einflußgrößen zu erweitern, z.B.:

- Wie wirken sich Zinserträge des rückgeflossenen Kapitals aus, oder
- wie wirken sich zusätzliche Aufwendungen während der Amortisationsdauer aus?

Rentabilität

Unter Rentabilität wird in der Betriebswirtschaftslehre das Verhältnis von Gewinn in einer Rechnungsperiode zu eingesetztem Kapital verstanden. Hierbei sind genau wie bei der Amortisationsrechnung verschiedene Ansätze möglich:

1. Unternehmensrentabilität

$$\frac{\text{Reingewinn} + \text{Fremdkapitalzinsen}}{\text{Unternehmenskapital}}$$

2. Rentabilität des Eigenkapitals

$$\frac{\text{Reingewinn}}{\text{Eigenkapital}}$$

3. Rentabilität des Betriebs

$$\frac{\text{Betriebsgewinn}}{\text{Betriebsnotwendiges Kapital}}$$

4. Umsatzrentabilität

$$\frac{\text{Gewinn}}{\text{Umsatz}}$$

5. Return of Investment (ROI)

$$\frac{\text{Gewinn}}{\text{Umsatz}} \times \frac{\text{Umsatz}}{\text{inv. Kapital}} = \frac{\text{Gewinn}}{\text{inv. Kapital}} \quad \text{(falls Gesamtansatz)}$$

In unserem Programm errechnen wir die allgemeine Rentabilität (durch entsprechende Interpretation der Daten werden alle oben aufgeführten Rentabilitätsansätze erfaßt) und den Return of Investment.

Beispiel

Nehmen wir wieder unser obiges Hühnerbeispiel. Wir mußten unseren Hühnern täglich eine Schüssel Wasser vorsetzen. Hiermit waren wir täglich etwa 1 Minute beschäftigt. Da wir privat nur ca. eine Stunde für solche Arbeiten zur Verfügung haben und wir in dieser Zeit eine kleine Nebentätigkeit ausführen wollen, geht uns diese Zeit dort verloren. Da wir für diese Nebentätigkeit 15,– DM bekommen, kostet uns die entgangene Einnahme also 0,25 DM (jährlich 365 x 0,25 DM = 91,25 DM). Wir beschließen deshalb, in eine Tränkanlage zu investieren, die automatisch aus der Wasserleitung nachgefüllt wird. Diese Anlage kostet mit allen Zusatzkosten 125,– DM. Durch die ständige Verfügbarkeit frischen Wassers erhöht sich jedoch die Legeleistung unserer Hühner, und wir erhöhen den Gewinn um 101,25 DM (91,25 DM + 10,– DM). Hierfür wollen wir die Rentabilität und den Return of Investment errechnen.

```
P53 – RENTABILITÄT
KAPITALEINSATZ              125,00
JAEHRL. GEWINN              101,25
JAEHRL. KOSTENERSP.          91,25

HIERAUS ERGIBT SICH EINE
RENTABILITÄT VON

81 %

UND EIN
RETURN OF INVESTMENT VON

73 %

WEITER, DANN J EINGEBEN
```

Erweiterungen

Auch hier sind einfache Erweiterungen möglich wie

- die Einbeziehung von Zinserträgen und
- die Formulierung eigener Rentabilitätskennzahlen

An den oben geschilderten einfachen Problemlösungen erkennen Sie, daß ein wesentlicher Anteil der Programme und damit auch des Erstel-

lungsaufwands von Programmen für die Steuerung des Programmablaufs und die Überprüfung von Fehleingaben aufgewendet werden muß. Ist erst einmal der Gesamtrahmen mit der Strukturierung der Haupt- und Teilmenüs sowie der Teilsteuerungsabläufe formuliert, so ist der Aufwand für die Codierung der eigentlichen Problemlösung von fast untergeordneter Bedeutung.

Nachdem Sie nun mit den wichtigsten Grundlagen der Programmierung und Anwendung Ihres TI-99/4A vertraut sind, empfehlen wir Ihnen weiterführende Übungen und Anwendungen. Auch können Sie die im Handel erhältlichen Programme nun leicht an Ihre speziellen Anforderungen anpassen. Wir wünschen Ihnen hierzu viel Erfolg.

Anhang A

Die ASCII-Code-Tabelle

ASCII-Code Tabelle

	dez	okt	hex		bin
NUL	0	000	00	CTRL @	0000 0000
SOH	1	001	01	CTRL A	0000 0001
STX	2	002	02	CTRL B	0000 0010
ETX	3	003	03	CTRL C	0000 0011
EOT	4	004	04	CTRL D	0000 0100
ENQ	5	005	05	CTRL E	0000 0101
ACK	6	006	06	CTRL F	0000 0110
BEL	7	007	07	CTRL G	0000 0111
BS	8	010	08	CTRL H	0000 1000
HT	9	011	09	CTRL I	0000 1001
LF	10	012	0A	CTRL J	0000 1010
VT	11	013	0B	CTRL K	0000 1011
FF	12	014	0C	CTRL L	0000 1100
CR	13	015	0D	CTRL M	0000 1101
SO	14	016	0E	CTRL N	0000 1110
SI	15	017	0F	CTRL O	0000 1111
DLE	16	020	10	CTRL P	0001 0000
DC1	17	021	11	CTRL Q	0001 0001
DC2	18	022	12	CTRL R	0001 0010
DC3	19	023	13	CTRL S	0001 0011
DC4	20	024	14	CTRL T	0001 0100
NAK	21	025	15	CTRL U	0001 0101
SYN	22	026	16	CTRL V	0001 0110
ETB	23	027	17	CTRL W	0001 0111
CAN	24	030	18	CTRL X	0001 1000
EM	25	031	19	CTRL Y	0001 1001
SUB	26	032	1A	CTRL Z	0001 1010
ESC	27	033	1B	CTRL [	0001 1011
FS	28	034	1C	CTRL \	0001 1100
GS	29	035	1D	CTRL]	0001 1101
RS	30	036	1E	CTRL ^	0001 1110
US	31	037	1F	CTRL -	0001 1111

	dez	okt	hex		bin
	32	040	20		0010 0000
!	33	041	21	!	0010 0001
"	34	042	22	"	0010 0010
#	35	043	23	#	0010 0011
$	36	044	24	$	0010 0100
%	37	045	25	%	0010 0101
&	38	046	26	&	0010 0110
'	39	047	27	'	0010 0111
(	40	050	28	(	0010 1000
)	41	051	29	)	0010 1001
*	42	052	2A	*	0010 1010
+	43	053	2B	+	0010 1011
,	44	054	2C	,	0010 1100
-	45	055	2D	-	0010 1101
.	46	056	2E	.	0010 1110
/	47	057	2F	/	0010 1111
0	48	060	30	0	0011 0000
1	49	061	31	1	0011 0001
2	50	062	32	2	0011 0010
3	51	063	33	3	0011 0011
4	52	064	34	4	0011 0100
5	53	065	35	5	0011 0101
6	54	066	36	6	0011 0110
7	55	067	37	7	0011 0111
8	56	070	38	8	0011 1000
9	57	071	39	9	0011 1001
:	58	072	3A	:	0011 1010
;	59	073	3B	;	0011 1011
<	60	074	3C	<	0011 1100
=	61	075	3D	=	0011 1101
>	62	076	3E	>	0011 1110
?	63	077	3F	?	0011 1111

	dez	okt	hex		bin
@	64	100	40	@	0100 0000
A	65	101	41	A	0100 0001
B	66	102	42	B	0100 0010
C	67	103	43	C	0100 0011
D	68	104	44	D	0100 0100
E	69	105	45	E	0100 0101
F	70	106	46	F	0100 0110
G	71	107	47	G	0100 0111
H	72	110	48	H	0100 1000
I	73	111	49	I	0100 1001
J	74	112	4A	J	0100 1010
K	75	113	4B	K	0100 1011
L	76	114	4C	L	0100 1100
M	77	115	4D	M	0100 1101
N	78	116	4E	N	0100 1110
O	79	117	4F	O	0100 1111
P	80	120	50	P	0101 0000
Q	81	121	51	Q	0101 0001
R	82	122	52	R	0101 0010
S	83	123	53	S	0101 0011
T	84	124	54	T	0101 0100
U	85	125	55	U	0101 0101
V	86	126	56	V	0101 0110
W	87	127	57	W	0101 0111
X	88	130	58	X	0101 1000
Y	89	131	59	Y	0101 1001
Z	90	132	5A	Z	0101 1010
[	91	133	5B	[	0101 1011
\	92	134	5C	\	0101 1100
]	93	135	5D	]	0101 1101
^	94	136	5E	^	0101 1110
_	95	137	5F	_	0101 1111

	dez	okt	hex		bin
`	96	140	60	`	0110 0000
a	97	141	61	a	0110 0001
b	98	142	62	b	0110 0010
c	99	143	63	c	0110 0011
d	100	144	64	d	0110 0100
e	101	145	65	e	0110 0101
f	102	146	66	f	0110 0110
g	103	147	67	g	0110 0111
h	104	150	68	h	0110 1000
i	105	151	69	i	0110 1001
j	106	152	6A	j	0110 1010
k	107	153	6B	k	0110 1011
l	108	154	6C	l	0110 1100
m	109	155	6D	m	0110 1101
n	110	156	6E	n	0110 1110
o	111	157	6F	o	0110 1111
p	112	160	70	p	0111 0000
q	113	161	71	q	0111 0001
r	114	162	72	r	0111 0010
s	115	163	73	s	0111 0011
t	116	164	74	t	0111 0100
u	117	165	75	u	0111 0101
v	118	166	76	v	0111 0110
w	119	167	77	w	0111 0111
x	120	170	78	x	0111 1000
y	121	171	79	y	0111 1001
z	122	172	7A	z	0111 1010
{	123	173	7B	{	0111 1011
|	124	174	7C	|	0111 1100
}	125	175	7D	}	0111 1101
~	126	176	7E	~	0111 1110
DEL	127	177	7F	DEL	0111 1111

Anhang B

Zeichengruppen

Zeichengruppen-Nummer TI-BASIC	EXTENDED BASIC	Zeichencode
	0	30– 31
1	1	32– 39
2	2	40– 47
3	3	48– 55
4	4	56– 63
5	5	64– 71
6	6	72– 79
7	7	80– 87
8	8	88– 95
9	9	96–103
10	10	104–111
11	11	112–119
12	12	120–127
13	13	128–135
14	14	136–143
15		144–151
16		152–159

Farbcodierung

Um Vorder- und Hintergrundfarbe des Bildschirms mit Hilfe der COLOR-, SCREEN- und SPRITE-Unterprogramme bestimmen zu können, benötigt man die folgenden Farbcodes:

Farbcode	Farbe
1	transparent
2	schwarz
3	mittelgrün
4	hellgrün
5	dunkelblau
6	hellblau
7	dunkelrot
8	kornblumenblau
9	mittelrot
10	hellrot
11	dunkelgelb
12	hellgelb
13	dunkelgrün
14	magentarot
15	grau
16	weiß

Anhang C

Mustercodierung

Um Zeichen mit Hilfe des CHAR-Unterprogrammes selbst definieren zu können, benötigt man einen Muster-Code, der sich aus den Codezeichen für die folgenden Blöcke zusammensetzt:

BLOCK	Binär	Hexadezimal
□□□□	0000	0
□□□■	0001	1
□□■□	0010	2
□□■■	0011	3
□■□□	0100	4
□■□■	0101	5
□■■□	0110	6
□■■■	0111	7
■□□□	1000	8
■□□■	1001	9
■□■□	1010	A
■□■■	1011	B
■■□□	1100	C
■■□■	1101	D
■■■□	1110	E
■■■■	1111	F

Anhang D

Mustercodierung für den Standard-Zeichensatz
(Character Patterns)

Code	Zeichen	Pattern
32		0000000000000000
33	!	0010101010100010
34	"	0028282800000000
35	#	0028287C287C2828
36	$	0038545038145438
37	%	0060640810204C0C
38	&	0020505020544834
39	'	0008081000000000
40	(	0008102020201008
41	)	0020100808081020
42	*	000028107C102800
43	+	000010107C101000
44	,	0000000000301020
45	-	000000007C000000
46	.	0000000000003030
47	/	0000040810204000
48	0	0038444444444438
49	1	0010301010101038
50	2	003844040810207C
51	3	0038440418044438
52	4	00081828487C0808
53	5	007C407804044438
54	6	0018204078444438
55	7	007C040810202020
56	8	0038444438444438
57	9	003844443C040830
58	:	0000303000303000
59	;	0000303000301020
60	<	0008102040201008
61	=	0000007C007C0000
62	>	0020100804081020
63	?	0038440408100010
64	@	0038445C545C4038
65	A	003844447C444444
66	B	0078242438242478
67	C	0038444040404438
68	D	0078242424242478
69	E	007C40407840407C
70	F	007C404078404040
71	G	003C40405C444438
72	H	004444447C444444
73	I	0038101010101038
74	J	0004040404044438
75	K	0044485060504844
76	L	004040404040407C
77	M	00446C5454444444
78	N	00446464544C4C44
79	O	007C44444444447C
80	P	0078444478404040
81	Q	0038444444544834
82	R	0078444478504844
83	S	0038444038044438
84	T	007C101010101010
85	U	0044444444444438
86	V	0044444428281010
87	W	0044444454545428
88	X	0044442810284444
89	Y	0044442810101010
90	Z	007C04081020407C
91	[	0038202020202038
92	\	0000402010080400
93	]	0038080808080838
94	^	0000102844000000

Code	Zeichen	Pattern
95	_	000000000000007C
96	`	0000201008000000
97	a	00000038447C4444
98	b	0000007824382478
99	c	0000003C4040403C
100	d	0000007824242478
101	e	0000007C4078407C
102	f	0000007C40784040
103	g	0000003C405C4438
104	h	00000044447C4444
105	i	0000003810101038
106	j	0000000808084830
107	k	0000002428302824
108	l	000000404040407C
109	m	000000446C544444
110	n	0000004464544C44
111	o	0000007C4444447C
112	p	0000007844784040
113	q	0000003844544834
114	r	0000007844784844
115	s	0000003C40380478
116	t	0000007C10101010
117	u	0000004444444438
118	v	0000004444282810
119	w	0000004444545428
120	x	0000004428102844
121	y	0000004428101010
122	z	0000007C0810207C
123	{	0018202040202018
124	|	0010101000101010
125	}	0030080804080830
126	‾	0000205408000000
127		0000000000000000
128		0000000000000000
129		0000000000000000
130		0000000000000000

Anhang E

MASCHINENCODES DES TMS 9900

MNEMONIC	OPCODE	FORMAT	BEFEHL
A	A000	1	ADDITION
AB	B000	1	ADD BYTE
ABS	0740	6	ABSOLUTE VALUE
AI	0220	8	ADD IMMEDIATE
ANDI	0240	8	AND IMMEDIATE
B	0440	6	BRANCH
BL	0680	6	BRANCH AND LINK
BLWP	0400	6	BRANCH AND LOAD WORK SPACE POINTER
C	8000	1	COMPARE WORD
CB	9000	1	COMPARE BYTE
CI	0280	8	COMPARE IMMEDIATE
CKOF	03C0	7	CONTROL COMMAND
CKON	03A0	7	CONTROL COMMAND
CLR	04C0	6	CLEAR
COC	2000	3	COMPARE ONES CORRESPONDING
CZC	2400	3	COMPARE ZEROS CORRESPONDING
DEC	0600	6	DECREMENT
DECT	0640	6	DECREMENT BY TWO
DIV	3C00	9	DIVIDE
IDLE	0340	7	CONTROL COMMAND
INC	0580	6	INCREMENT
INCT	05C0	6	INCREMENT BY TWO
INV	0540	6	INVERT
JEQ	1300	2	JUMP EQUAL
JGT	1500	2	JUMP GREATER THEN ARITHMETIC
JH	1B00	2	JUMP HIGH LOGIC
JHE	1400	2	JUMP HIGH OR LESS LOGIC
JL	1A00	2	JUMP LESS LOGIC

MNEMONIC	OPCODE	FORMAT	BEFEHL
JLE	1200	2	JUMP LESS OR EQUAL LOGIC
JLT	1100	2	JUMP LESS THEN ARITHMETIC
JMP	1000	2	UNCONDITIONAL JUMP
JNC	1700	2	JUMP NO CARRY
JNE	1600	2	JUMP NOT EQUAL
JNO	1900	2	JUMP NO OVERFLOW
JOC	1800	2	JUMP ON CARRY
JOP	1C00	2	JUMP ODD PARITY
LDCR	3000	4	LOAD CRU
LI	0200	8	LOAD IMMEDIATE
LIMI	0300	8	LOAD INTERRUPT MASK IMMEDIATE
LREX	03E0	7	CONTROL COMMAND
LWPI	02E0	8	LOAD WORKSPACE POINTER IMMEDIATE
MOV	C000	1	MOVE WORD
MOVB	D000	1	MOVE BYTE
MPY	3800	9	MULTIPLY
NEG	0500	6	NEGATE
ORI	0260	8	OR IMMEDIATE
RSET	0360	7	CONTROL COMMAND
RTWP	0380	7	RETURN WITH WORKSPACE POINTER
S	6000	1	SUBTRACT WORDS
SB	7000	1	SUBTRACT BYTES
SBO	1D00	2	SET CRU BIT TO ONE
SBZ	1E00	2	SET CRU BIT TO ZERO
SETO	0700	6	SET TO ONE
SLA	0A00	5	SHIFT LEFT ARITHMETIC
SOC	E000	1	SET ONES CORRESPONDING
SOCB	F000	1	SET ONES CORRESPONDING BYTE
SRA	0800	5	SHIFT RIGHT ARITHMETIC
SRC	0800	5	SHIFT RIGHT CIRCULAR
SRL	0900	5	SHIFT RIGHT LOGICAL
STCR	3400	4	STORE CRU
STST	02C0	8	STORE STATUS
STWP	02A0	8	STORE WORKSPACE POINTER
SWPB	06C0	6	SWAP BYTES
SZC	4000	1	SET ZEROS CORRESPONDING
SZCB	5000	1	SET ZEROS CORRESPONDING BYTE
TB	1F00	2	TEST CRU BIT
X	0480	6	EXECUTE
XOP	2C00	9	EXTENDED OPERATION
XOR	2800	3	EXCLUSIVE OR

BEFEHLSFORMATE:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 Arithmetic	S				Ts		D				Td		B	OP-CODE		
2 JUMP	relative Sprungadresse								OP-CODE							
3 LOGIC	S				Ts		D				OP-CODE					
4 CRU	S				Ts		Zähler				OP-CODE					
5 SHIFT	R				Zähler				OP-CODE							
6 OPERATIONS	S				Ts		OP-CODE									
7 CONTROL	0	0	0	0	0	OP-CODE										
8 IMMEDIATE	R				0	OP-CODE										
9 MPY, DIV, XOP	S				Ts		D				OP-CODE					

S — Source (Speicherstelle, die als Quelle dient)

Ts — Adressierungsart der Quellenadresse
00 : Register
01 : Indirekt
10 : Index
11 : Indirekt mit automat. Erhöhung des Quellwertes

D — Destination (Zieladresse)

Td — Adressierungsart der Zieladresse
00 bis 10 : wie bei Ts
11 : Indirekt mit automat. Erhöhung des Zielwertes

R — Register (R0 bis RF)

B — BYTE Bit (B = 1: Byte Befehl, B = 0: Wort Befehl)

OP-CODE — Operations Code

Zähler — Anzahl der Wiederholungen

Referenzliste OP-Code / Mnemonic:

0200	LI
0220	AI
0240	ANDI
0260	ORI
0280	CI
02A0	STWP
02C0	STST
02E0	LWPI
0300	LIMI
0340	IDLE
0360	RSET
0380	RTWP
03A0	CKON
03C0	CKOF
03E0	LREX
0400	BLWP
0440	B
0480	X
04C0	CLR
0500	NEG
0540	INV
0580	INC
05C0	INCT
0600	DEC
0640	DECT
0680	BL
06C0	SWPB
0700	SETO
0740	ABS
0800	SRA
0800	SRC
0900	SRL
0A00	SLA
1000	JMP
1100	JLT
1200	JLE
1300	JEQ
1400	JHE
1500	JGT
1600	JNE
1700	JNC
1800	JOC
1900	JNO
1A00	JL
1B00	JH
1C00	JOP
1D00	SBO
1E00	SBZ
1F00	TB
2000	COC
2400	CZC
2800	XOR
2C00	XOP
3000	LDCR
3400	STCR
3800	MPY
3C00	DIV
4000	SZC
5000	SZCB
6000	S
7000	SB
8000	C
9000	CB
A000	A
B000	AB
C000	MOV
D000	MOVB
E000	SOC
F000	SOCB

Anhang F

Speicherbelegung des CPU RAM des TI-99/4A

(Alle Adressen in hexadezimal)

ROM IN DER KONSOLE

0000 - 003E	Interrupt Vektoren	
0040 - 1FFF	Konsole ROM Routinen, Betriebssystem BASIC Interpreter	

SPEICHERERWEITERUNG

2000 - 3FFF	Speichererweiterung 8 KB Segment	

PERIPHERIE ROM

4000 - 5FFF	Peripherie ROM. Serviceroutinen der Peripherieeinheiten	

MINIMEM ROM SEGMENT

6000	ROM Kopfzeile	
6010	XML >70	Routine für Programmnamensuche
6012	XML >71	Maschinencode Ladeprogramm
6014	XML >72	CIF nicht benützt
6016		nicht benützt
6018	aGPLLINK	Verbindung zur GROM Routine
6019	aXMLLNK	Verbindung zur ROM Routine
6020	aKSCAN	Verbindung zur Tastatur Routine
6024	aVSBW	Byte CPU RAM -> VDP RAM
6028	aVMBW	mehrere Bytes CPU RAM -> VDP RAM
602C	aVSBR	Byte VDP RAM -> CPU RAM
6030	aVMBR	mehrere Bytes VDP RAM -> CPU RAM
6034	VWTR	Schreiben von VDP Register
6038	aDSRLNK	Verbindung zur DSR
603C	aLOADER	Maschinenprogrammlader
6040	NUMASG	Assemler Wert -> BASIC Wert
6044	aNUMREF	BASIC Wert -> Assembler Wert
6048	aSTRASG	Assembler String -> BASIC String
604C	aSTRREF	BASIC String -> Assemblerstring
6050	aERR	Fehlercode Assembler -> BASIC
6054 - 6FFF		Programmteil

MINIMEM RAM SEGMENT mit BASIC FELDERN

7000	Kennwort >5AA5
7002	Dateiinformation, Status
7003	Satzlänge
7004	Dateiende Zeiger
7006	Dateianfangsadresse
7008 - 7FFF	Dateifelder

MINIMEM RAM SEGMENT mit ASSEMBLERPROGRAMMEN

7000	Kennwort >5AA5
701C	erste freie Adresse im MINIMEM
701E	letzte freie Adresse im MINIMEM
7020	Startadresse >0000
7022	erste freie Adresse im oberen Speicherbereich
7024	letzte freie Adresse im oberen Speicherbereich
7026	erste freie Adresse im unteren Speicherbereich
7028	letzte freie Adresse im unteren Speicherbereich
702A	Prüfsummen Wert
702C	PAB-Zeiger
702E	GPL-Rücksprungadresse
7030	CRU-Adresse der Peripherie
7034	Namenlänge der Peripherieeinheit
7036	Zeiger auf Namen der Peripherie im PAB
7038	DSR-Nummer
703A	80 Byte Buffer für Rekorder
708A	NAMEN Buffer
7092	Arbeitsbereich für Routinen
7098	DSR-Arbeitsbereich
70B8	Anwender-Arbeitsbereich
70D8	Loader-Arbeitsbereich
70F8	Interner Datenbereich
7118 - 7FFF	freier Platz für Programme und Daten
7FFF	Beginn der Liste der Programmnamen

UNTERPROGRAMME FÜR VDP, TON UND SPRACHE

8000 - 82FF	

SCRATCH PAD RAM

8300	- 8349	wird bei CALL LINK verwendet
834A	- 836D	wird als Wertebuffer benötigt
836E	- 836F	Buffer für BASIC-Gleitkommaberechnungen
8370	- 837F	GPL Status Block
8370		höchste verfügbare Adresse im VDP RAM
8372		signifikantes Byte = hex 83
		nicht signifikantes Byte = Wertestapelzeiger
8373		Wertestapelzeiger der Routinen
8374		Tastaturmodus, normal = 0
8375		ASCII Code bei KSCAN Tastatur Routine
8376		Fernbedienung Y-Position
8377		Fernbedienung X-Position
8378		Zufallszahlengenerator
8379		VDP Interrupt Zähler, wird alle 1/60sek .erhöht
837A		Anzahl der in Bewegung befindlichen Sprites
837B		Kopie des VDP Status Byte
837C		GPL Status Byte
837D		Zeichenbuffer des VDP
837E		Bildschirm-Zeilenzeiger
837F		Bildschirm-Spaltenzeiger
8380	- 83BF	Unterprogramm Wertestapel
83C0	- 83DF	Interpreter Arbeitsbereich
83E0	- 83FF	GPL-Arbeitsbereich

Speichererweiterung 24 KB Segment

Anhang G

TI-99/4A Extended Basic Referenzkarte

Übersicht über Befehle, Anweisungen und Funktionen des TI-BASIC des TI-99/4A Heimcomputers. Weitergehende Beschreibungen sind dem Programmierhandbuch TI BASIC/Extended BASIC für Anfänger und Fortgeschrittene zu entnehmen.

C = Befehl (Command), **F** = Funktion, **S** = Anweisung (Statement)

ABS (*numerischer Ausdruck*)
ermittelt den Absolutwert eines numerischen Ausdrucks **F**

ACCEPT [[AT (*Zeile, Spalte*)] [VALIDATE (*Typbezeichnung, . . .*)]
[BEEP] [ERASE ALL] [SIZE (*numerischer Ausdruck*)]
:] *Variable* **S, C**
unterbricht die Programmausführung, bis über die Tastatur der Variablenwert eingegeben wurde. Bei diesem Befehl sind die in eckige Klammern gesetzten Optionen möglich:
AT (*Zeile, Spalte*) positioniert den Cursor für die Dateneingabe
VALIDATE (*Typbezeichnung . . .*) grenzt den Wertebereich für die Variable ein
UALPHA erlaubt nur Großbuchstaben,
DIGIT erlaubt nur die Ziffern 0 bis 9
NUMERIC erlaubt die Ziffern 0 bis 9 und ".", "+", "," und E
„String" erlaubt die Eingabe der angegebenen Zeichen
BEEP: erzeugt einen Ton als Bereitschaftssignal
ERASE ALL: löscht vor der Dateneingabe den gesamten Bildschirm
SIZE (*numerischer Ausdruck*): begrenzt die Zahl der eingegebenen Zeichen. Bei postivem Wert des numerischen Ausdrucks wird das Eingabefeld vor der Eingabe gelöscht, bei negativem Wert bleibt der alte Wert erhalten.

ASC (*Stringausdruck*)
ermittelt den entsprechenden ASCII-Code des ersten Buchstabens des Stringausdrucks. **F**

ATN (*numerischer Ausdruck*)
ermittelt den Winkel (im Bogenmaß), dessen Tangens den numerischen Ausdruck ergibt (arcus tangens). **F**

BREAK [*Zeilennummernliste*]
veranlaßt eine Programmunterbrechung bei Erkennen des Befehls (fehlende Zeilennummernliste) oder bei den in der Zeilennummernliste angegebenen Zeilen (bei besetzter Zeilennummernliste). **C, S**

BYE
schließt alle offenen Dateien und verläßt das BASIC. **C**

CALL *Unterprogramm* [(*Parameterliste*)]
ruft das angegebene Unterprogramm auf. Die eventuell angegebenen Parameter werden verarbeitet. **S**

CALL CHAR (*Zeichencode, „Mustercode“*)
definiert mit dem angegebenen Mustercode ein spezielles Zeichen innerhalb des ASCII-Code-Bereichs. Der Mustercode kann 0 bis 64 hexadezimale Ziffern lang sein. **S, C**

CALL CHARPAT (*Zeichencode, Stringvariable* [, . . .])
ermittelt den Hexadezimalcode eines mit CALL CHAR definierten Zeichens. **S, C**

CALL CHARSET
setzt die Zeichen im Bereich 32 bis 95 auf ihre ursprünglich vordefinierte Form (Muster) zurück. **S, C**

CHR$ (*numerischer Ausdruck*)
ermittelt das Zeichen, das dem ASCII-Wert des numerischen Ausdrucks entspricht. **F**

CALL CLEAR
löscht den Bildschirm, indem alle Bildschirmstellen mit Leerzeichen gefüllt werden. **S, C**

CLOSE # *Dateinummer* [: DELETE]
schließt die nach # angegebene Datei. Bei Angabe von DELETE wird außerdem die Datei gelöscht. **S, C**

CALL COINC (# *Spritenummer 1,* # *Spritenummer 2, Toleranz, numerische Variable*)

CALL COINC (# *Spritenummer, Punktzeile, Punktspalte, Toleranz, numerische Variable*)

CALL COINC (ALL, *numerische Variable*)
ermittelt das eventuelle Zusammenfallen zweier oder mehrerer Sprites in einer durch Punktzeile und Punktspalte bestimmten Bildschirmposition. Bei Angabe von ALL wird jedes Zusammenfallen zweier Sprites ermittelt. Das Zusammentreffen wird in der numerischen Variablen mit −1 angegeben, ein fehlendes Zusammentreffen wird durch die Ziffer 0 angezeigt. **S, C**

CALL COLOR (# *Spritenummer, Vordergrund-Farbcode* [, . . .])

CALL COLOR (*Zeichengruppennummer, Vordergrund-Farbcode, Hintergrund-Farbcode*)
legt die Vorder- oder Hintergrundfarben einzelner Sprites durch die Angabe eines Farbcodes fest. **S, C**

CONTINUE

CON
ermöglicht die definierte Weiterbearbeitung des Programms nach einer Unterbrechung durch BREAK bzw. durch die spezielle Tastenfunktion CLEAR (FCTN −4). **C**

COS (*numerischer Ausdruck für Bogenmaß*)
ermittelt den trigonometrischen Kosinus des mit dem numerischen Ausdruck bestimmten Bogenmaßes. **F**

DATA *Datenliste*
erlaubt die Abspeicherung von Daten innerhalb eines Programms. **S**

DEF *Funktionsname* [(*Variable*)] = *Ausdruck*
ordnet einen vom Benutzer definierten numerischen oder String-Ausdruck einem Funktionsnamen zu. **S**

DELETE *Gerätebezeichnung. Dateiname*
löscht Dateien auf externen Speichern. **C, S**

CALL DELSPRITE
(# Spritenummer [, ...])

CALL DELSPRITE (ALL)
löscht spezifizierte Sprites. Mit ALL werden alle Sprites im Bildschirmbereich gelöscht. **S, C**

DIM *Datenfeldname* (*ganze Zahl 1* [, *ganze Zahl 2*] [,. . .] [*ganze Zahl 7*])
[. . .]
dimensioniert Vektoren und Felder. **S, C**

DISPLAY [[AT (*Zeile, Spalte*)] [BEEP] [ERASE ALL]
[SIZE (*numerischer Ausdruck*)]:] *Variablenliste* **S, C**
ermöglicht eine Ausgabe von Daten an spezifizierten Bildschirmpunkten.
AT (*Zeile, Spalte*) positioniert das erste ausgegebene Zeichen an die angegebene Zeile und Spalte.
BEEP erzeugt einen Ton als Ausgabesignal.
ERASE ALL löscht vor der Ausgabe den gesamten Bildschirm.
SIZE (*numerischer Ausdruck*) gibt eine dem numerischen Wert entsprechende Anzahl von Leerzeichen aus.

DISPLAY [*Optionsliste* :] USING *Stringausdruck* [: *Variablenliste*]

DISPLAY [*Optionsliste* :] USING *Zeilennummer* [: Variablenliste]
hat die gleiche Funktion wie die einfache DISPLAY-Anweisung mit der Zusatzfunktion USING für den Formataufbau. Mit dem Stringausdruck wird das gewünschte Format definiert. Mit Angabe der Zeilennummer wird auf einen IMAGE-Befehl verzweigt (vgl. auch die Beschreibung des IMAGE-Befehls). **S, C**

CALL DISTANCE (# *Spritenummer*, # *Spritenummer, numerische Variable*)

CALL DISTANCE (# *Spritenummer, Punktzeile, Punktspalte, numerische Variable*)
ermittelt die Quadratzahl der Entfernung zwischen zwei Punkten bzw. zwischen einem Punkt und einer bestimmten Bildschirmposition. **S, C**

END
beendet die Programmausführung.

EOF (*Dateinummer*) **S**
prüft die Datei-Endbedingung der mit der Dateinummer angegebenen Datei. **F**
 0: Dateiende nicht erreicht
 1: logisches Dateiende
−1: physikalisches Dateiende

CALL ERR (*Fehlercode, Fehlertyp* [, *Fehlergewicht, Zeilennummer*]) **S, C**
ermittelt Fehlercode und Fehlertyp des letzten unbereinigten Fehlers.
Fehlercode: zwei- oder dreistellige Zahl (Bedeutung, vgl. Handbuch).
Fehlertyp: bei negativer Zahl wurde Fehler bei der Programmausführung erkannt. Bei positiver Zahl ergibt die Zahl die Dateinummer, bei der ein Fehler erkannt wurde.
Fehlergewicht: 9 zeigt an, daß der Fehler nicht behebbar ist.

EXP (*numerischer Ausdruck*)
ermittelt den Exponentialwert (e^x) eines numerischen Ausdrucks. Der Wert von e beträgt 2.718281828. **F**

FOR *Kontrollvariable* = *Startwert* TO *Endwert* [STEP *Schrittweite*]
wiederholt die Ausführung der Anweisungen zwischen FOR und NEXT, bis die Kontrollvariable den Endwert erreicht.
Der Standardwert für STEP ist 1. **S, C**

CALL G CHAR (*Zeile, Spalte, numerische Variable*)
ermittelt den ASCII-Code eines Zeichens an der durch Zeile und Spalte vorgegebenen Bildschirmposition und legt diesen in der numerischen Variablen ab. **S, C**

GOSUB *Zeilennummer*

GO SUB *Zeilennummer*
ist ein unbedingter Sprung zu dem bei der Zeilennummer beginnenden Unterprogramm. **S**

GOTO *Zeilennummer*

GO TO *Zeilennummer*
ist ein unbedingter Sprung zu der Zeilennummer. **S**

CALL HCHAR (*Zeile, Spalte, Zeichencode* [, *Wiederholungsanzahl*])
setzt das ASCII-Zeichen an die mit Zeile und Spalte angegebene Bildschirmposition. Mit der zusätzlich angegebenen Wiederholungsanzahl wird die Häufigkeit bestimmt, mit der dieses Zeichen wiederholt wird. **S, C**

IF *logischer Ausdruck* THEN *Zeilennummer 1* [ELSE *Zeilennummer 2*]

IF *logischer Ausdruck* THEN *Anweisung 1* [ELSE *Anweisung 2*]

IF *numerischer Ausdruck* THEN *Zeilennummer 1* [ELSE *Zeilennummer 2*]

IF *numerischer Ausdruck* THEN *Anweisung 1* [ELSE *Anweisung 2*]
übergibt die Programmsteuerung an Zeilennummer 1 bzw. führt Anweisung 1 aus, wenn der logische Ausdruck wahr oder der numerische Ausdruck ungleich Null ist. Im anderen Fall verzweigt das Programm auf Zeilennummer 2 bzw. führt die Anweisung 2 aus. **S**

IMAGE *Formatstring*
definiert das Format für die mit PRINT USING oder DISPLAY USING auszugebenden Daten. Ein Formatstring kann folgende Form bzw. Werte aufweisen:
- Buchstaben, Ziffern außer # und ∧ : direkte Ausgabe.
- #: Stellvertreter für das Format der auszugebenden Zahl.
- ∧: Stellvertreter für die Ausgabe von Exponentialzahlen. **S**

CALL INIT
bereitet den Computer auf die Ausführung von Assembler-Unterprogrammen vor. **S, C**

INPUT [*Eingabe-Dialogzeile* :] *Variablenliste*
unterbricht die Programmausführung, bis über die Tastatur eine Eingabe erfolgt ist. Eine zusätzlich angegebene Zeichenkette als Dialogzeile wird vor der Eingabe auf den Bildschirm geschrieben. **S**

INPUT # *Zeilennummer* [. REC *Satznummer*]: *Variablenliste*
ordnet Daten aus einer Datei den Variablen einer Variablenliste zu. Ohne REC-Klausel werden die Sätze sequentiell gelesen. **S**

INT (*numerischer Ausdruck*)
ermittelt die größte ganze Zahl, die kleiner oder gleich dem numerischen Ausdruck ist. **F**

CALL JOYST (*Tastaturmodus, x-Wert, y-Wert*)
übernimmt die Joystickposition als x- und y-Wert. Die Werte sind −4,0 und 4. **S, C**

CALL KEY (*Tastaturmodus, Rückmeldevariable, Statusvariable*)
ermöglicht die direkte Eingabe von Tastaturzeichen ohne Eingabe von ENTER. An die Rückmeldevariable wird dabei der Code der gedruckten Taste übergeben. In der Statusvariablen wird dabei der Tastaturstatus übernommen

1: es wurde eine neue Taste gedrückt,
−1: es wurde die gleiche Taste nochmals gedrückt,
0: es wurde im Moment der Ausführung keine Taste gedrückt. **S, C**

LEN (*Stringausdruck*)
ermittelt die Länge eines Stringausdrucks (Anzahl der Zeichen). **F**

[LET] *numerische Variable* [, *numerische Variable* ,. . .] = *numerischer Ausdruck*

[LET] *Stringvariable* [, *Stringvariable* ,. . .] = *Stringausdruck*
ordnet den Wert eines Ausdrucks einer (mehreren) Variablen zu. **S, C**

CALL LINK (*Unterprogrammname* [, *Variablenliste*])
übergibt die Kontrolle und, falls vorhanden, eine Variablenliste an ein Assembler-Unterprogramm. **S, C**

LINPUT [[# *Dateinummer*] [, REC *Satznummer*]:] *Stringvariable*
ordnet Daten aus einer Datei einer Stringvariablen zu. Bei Angabe der REC-Klausel wird direkt die angegebene Satznummer angesprochen. **S**

LINPUT [*Eingabe-Dialogzeile* :] *Stringvariable*
unterbricht die Programmausführung, bis eine vollständige Eingabezeile eingegeben wurde. Hierbei werden auch Sonderzeichen akzeptiert. **S**

LIST [„*Gerätename*“:] [*Zeilennummernliste*]

LIST [„*Gerätename*“:] [*Startzeilennummer*] – [*Endzeilennummer*]
gibt sequentiell alle Programmzeilen oder bei Angabe der Start- und Endzeilennummer die Anweisungen innerhalb dieser Zeilen aus. **C**

CALL LOAD (“*Dateibezeichnung*“ [, *Adresse, Byte*] [,. . .] [, *Dateifeld*] [,. . .])
lädt ein Assembler-Unterprogramm. **S, C**

CALL LOCATE (# *Spritennummer, Punktzeile, Punktspalte* [,. . .])
positioniert den angegebenen Sprite an die mit Punktzeile und Punktspalte angegebene Bildschirmposition. **S, C**

LOG (*numerischer Ausdruck*)
ermittelt den natürlichen Logarithmus des numerischen Ausdrucks. **F**

CALL MAGNIFY (*Vergrößerungsfaktor*)
legt Größe und Anzahl der einen Sprite bildenden Punkte fest.

Vergrößerungsfaktor:
1: einfache Größe, 1 Zeichen
2: doppelte Größe, 1 Zeichen
3: einfache Größe, 4 Zeichen
4: doppelte Größe, 4 Zeichen

MAX (*numerischer Ausdruck 1, numerischer Ausdruck 2*)
ermittelt den größeren der beiden numerischen Ausdrücke 1 und 2. **F**

MERGE [“] *Gerätename . Programmname* [“]
übernimmt das mit Gerätename . Programmname spezifizierte Programm in den Arbeitsspeicher. **C**

MIN (*numerischer Ausdruck 1, numerischer Ausdruck 2*)
ermittelt den kleineren der numerischen Ausdrücke 1 und 2. **F**

CALL MOTION (# *Spritenummer, Zeilengeschwindigkeit, Spaltengeschwindigkeit* [,. . .])
stellt die Geschwindigkeiten in Zeilen- und Spaltenrichtung eines oder mehrerer Sprites ein. **S, C**

NEW
löscht den Programm- und Bildschirmspeicher und bereitet so die Eingabe eines neuen Programms vor. **C**

NEXT *Kontrollvariable*
vgl. auch FOR-Anweisung. **S, C**

NUMBER [*Startzeile*] [,*Schrittweite*]

NUM [*Startzeile*] [,*Schrittweite*]
generiert Zeilennummern für die Programmeingabe. Ohne Angabe von Startzeile und Schrittweite wird die Startzeile auf 100 und die Schrittweite auf 10 gesetzt. **C**

OLD Gerätename [.*Programmname*]
lädt ein Programm vom externen Datenspeicher in den Programmspeicher. **C**

ON BREAK STOP

ON BREAK NEXT
bestimmt die Art der Behandlung eines BREAK-Befehls bzw. von von **FCTN-4** (CLEAR). Ohne weitere Angabe wird die STOP-Anweisung ausgeführt. Mit NEXT wird mit der Programmausführung fortgefahren. **S**

ON ERROR STOP

ON ERROR *Zeilennummer*
bestimmt die Art der Fehlerbehandlung. Ohne weitere Angabe wird die STOP-Anweisung ausgeführt. Falls Zeilennummer angegeben wurde, verzweigt das Programm bei auftretendem Fehler dorthin. **S**

ON *numerischer Ausdruck* **GOSUB** *Zeilennummer* [,. . .]

ON *numerischer Ausdruck* **GO SUB** *Zeilennummer* [,. . .]
übergibt abhängig vom Wert des numerischen Ausdrucks die Kontrolle an ein Unterprogramm. Hierbei korrespondiert jeweils die Position des erfüllten numerischen Ausdrucks mit der Position der Startzeile des Unterprogramms.

ON *numerischer Ausdruck* **GOTO** *Zeilennummer* [,. . .]

ON *numerischer Ausdruck* **GO TO** *Zeilennummer* [,. . .]
übergibt abhängig vom Wert des numerischen Ausdrucks die Kontrolle an die Zeilennummer, deren Position mit der Position des numerischen Ausdrucks korrespondiert.

ON WARNING PRINT

ON WARNING STOP

ON WARNING NEXT
bestimmt die Art der Fehlerbehandlung bei Erkennen eines leichten Fehlers. Ohne weitere Angabe hinter WARNING wird der PRINT-Befehl ausgeführt, der die Fehlermeldung anzeigt. Mit STOP wird eine Programmunterbrechung durchgeführt und mit NEXT das Programm fortgesetzt. **S**

OPEN # *Dateinummer : "Dateibezeichnung"*
[. *Dateiorganisation*] [.*Datentyp*] [.*Eröffnungsmodus*] [.Datensatztyp]
bereitet die Benutzung einer externen Datei vor. Hierzu gehört z. B. der Kassettenrecorder, ein Diskettenlaufwerk oder ein Drucker.
Dateinummer: 0–255
Dateibezeichnung: Gerätename [. Dateiname]
Dateiorganisation: RELATIVE oder SEQUENTIAL
Datentyp: DISPLAY oder INTERNAL
Eröffnungsmodus
Datensatztyp: FIXED oder VARIABLE **S, C**

OPTION BASE 0

OPTION BASE 1
setzt den kleinsten erlaubten Index eines Datenfeldes auf 0 oder 1. Ohne Angabe wird er auf 0 gesetzt. **S**

CALL PATTERN (# *Spritenummer, Zeichencode* [,. . .])
verändert das Muster des oder der spezifizierten Sprites in das dem angegebenen Zeichencode entsprechende. **S, C**

CALL PEEK (*Adresse, numerische Variablenliste*)
ermittelt den Inhalt der angegebenen Speicheradresse. **S, C**

PI
gibt den Wert für $\pi = 3.14159265359$ zurück. **F**

POS (*String 1, String 2, numerischer Ausdruck*)
ermittelt die Position des ersten Zeichens von String 2 im String 1. Hierbei beginnt die Suche nach String 2 in String 1 ab der durch den numerischen Ausdruck bestimmten Position. Wird String 2 nicht gefunden, so wird 0 zurückgegeben. **F**

CALL POSITION (# *Spritenummer, Punktzeile, Punktspalte* [,. . .])
ermittelt die Bildschirmposition eines oder mehrerer Sprites. **S, C**

PRINT [# *Dateinummer* [, REC *Satznummer*]:] [*Ausgabeliste*]
gibt die Werte der in der Ausgabeliste definierten Variablen auf dem Bildschirm bzw. bei Angabe einer Dateinummer auf die entsprechende Datei aus. Mit der REC-Klausel und der Satznummer wird direkt der gewünschte Satz angesprochen. **S, C**

PRINT [# *Dateinummer* [, REC *Satznummer*]] **USING** *Stringausdruck: Ausgabeliste*

PRINT [# *Dateinummer* [, REC *Satznummer*]] **USING** *Stringausdruck: Ausgabeliste*
arbeitet wie PRINT, jedoch mit der zusätzlichen Möglichkeit der formatierten Ausgabe. Mit dem Stringausdruck wird das Format definiert. Mit der Zeilennummer wird auf eine dort stehende IMAGE-Anweisung verwiesen. **S, C**

RANDOMIZE [*numerischer Ausdruck*]
erzeugt eine nicht vorhersagbare Folge von Zufallszahlen. Durch den optionalen numerischen Ausdruck kann die Folge wiederholt werden.

READ *Variablenliste*
ordnet die in der DATA-Anweisung stehenden Daten sequentiell den Variablen in der Variablenliste zu. **S, C**

REC (*Dateinummer*)
ermittelt die aktuelle Satznummer. **F**

REM
steht für die interne Programmdokumentation zur Verfügung (keine Programmwirkung).

RESEQUENCE [*Anfangszeile*] [, *Schrittweite*] **S, C**

RES [*Anfangszeile*] [, *Schrittweite*]
bewirkt eine automatische Neuzuordnung der Zeilennummern. Ohne weitere Angaben wird der Wert der Anfangszeile auf 100 und die Schrittweite auf 10 gesetzt. **C**

RESTORE [*Zeilennummer*]
setzt den Zeiger für die nächste READ-Anweisung auf die Eintragungen der ersten DATA-Anweisung. Bei Angabe einer Zeilennummer wird der Zeiger auf die erste Data-Anweisung nach der angegebenen Zeilennummer gesetzt. **S, C**

RESTORE # *Dateinummer* [, REC *Satznummer*]
Setzt den Zeiger für die nächste Leseanweisung auf den Beginn der Datei bzw. bei Angabe einer Satznummer auf den entsprechenden Satz. **S, C**

RETURN
gibt die Steuerung vom Unterprogramm an das aufrufende Programm zurück (hinter GOSUB bzw. ON ... GOSUB). **S**

RETURN [*Zeilennummer*] **S**

RETURN [NEXT]
übergibt im Zusammenhang mit dem ON ERROR-Befehl im Falle eines erkannten Fehlers die Steuerung entweder an die Anweisung in der angegebenen Zeilennummer bzw. bei NEXT an die folgende Programmzeile. **S**

RND
erzeugt eine Pseudo-Zufallszahl im Bereich von 0 bis 1. **F**

RPT$ (*Stringausdruck, numerischer Ausdruck*)
wiederholt den Wert des Stringausdrucks entsprechend oft dem Wert des numerischen Ausdrucks.

RUN [*„Gerätename. Programmname“]*

RUN *Zeilennummer*
startet die Programmausführung, entweder vom Beginn bzw. der angegebenen Zeilennummer eines im Arbeitsspeicher befindlichen Programms aus, oder ein Programm mit dem angegebenen Programmnamen auf einem externen Datenträger. **C, S**

SAVE *Gerätename . Programmname* [PROTECTED]

SAVE *Gerätename . Programmname* [, MERGE]
kopiert das Programm im internen Arbeitsspeicher auf einen externen Datenträger. Mit PROTECTED wird ein späteres Ändern oder Listen verhindert. Mit MERGE wird ein Einbinden in ein anderes Programm ermöglicht.

CALL SAY (*Wortstring* [*Wiedergabestring*] [,. . .]) **C**
erzeugt die Ausgabe eines Wortstrings bzw. des Wertes aus dem Wiedergabestring auf dem Sprachsynthesizer. **S, C**

CALL SCREEN (*Farbcode*)
ändert die Bildschirmgrundfarbe. **S, C**

SEG$ (*Stringausdruck, Position, Länge*)
ermittelt den Teilstring der sich mit der angegebenen Länge ab der angegebenen Position aus einem Stringausdruck ergibt. **F**

SGN (numerischer Ausdruck)
ermittelt das algebraische Vorzeichen eines numerischen Ausdrucks. Es ergibt sich 1 für einen positiven, 0 für Null und −1 für einen negativen Wert. **F**

SIN (*numerischer Ausdruck im Bogenmaß*)
ermittelt den trigonometrischen Sinuswert eines numerischen Ausdrucks im Bogenmaß. **F**

SIZE
zeigt die Größe des noch frei verfügbaren Arbeitsspeichers an. **C**

CALL SOUND (*Dauer, Frequenz 1, Lautstärke 1* [,. . .] [*Frequenz 4, Lautstärke 4*])
steuert ein bis drei Tongeneratoren und einen Rauschgenerator. Ton- und Rauschgenerator können in beliebiger Reihenfolge aufgerufen werden. Bei negativer Dauer werden eventuell vorher angestoßene Töne bzw. Geräusche vorzeitig abgebrochen.
Dauer: 1 bis 4250 ms, −4250 bis −1 ms
Frequenz: 110 bis 44733 Hz für Töne, −1 bis −8 für Rauschen (Geräusche)
Lautstärke: 0 (lauteste Stärke) bis 30 (leiseste Stärke) **S, C**

CALL SPGET (*Wortstring, Wiedergabestring*)
ermittelt das Sprachmuster des Wortstrings und ordnet diesen dem Wiedergabestring zu. **S, C**

CALL SPRITE (# *Spritenummer, Zeichencode, Farbcode, Punktzeile, Punktspalte* [, *Zeilengeschwindigkeit, Spaltengeschwindigkeit*] [,. . .])
definiert einen mit # Spritenummer erklärten Sprite durch die Angabe des gewünschten Zeichencode, Farbcode am Ort Punktzeile und Punktspalte. Mit zusätzlich angegebener Zeilen- und Spaltengeschwindigkeit werden außerdem die Geschwindigkeitswerte gesetzt. **S, C**

SQR (*numerischer Ausdruck*)
ermittelt die Quadratwurzel ($\sqrt{\ }$) des numerischen Ausdrucks. **S, C**

STR$ (*numerischer Ausdruck*)
wandelt den Wert des numerischen Ausdrucks in einen Stringausdruck um. **F**

SUB *Unterprogrammname* [(*Variablenliste*)]
zeigt den Beginn eines Unterprogramms an. Die Variablenliste zeigt die zu übergebenden Variablen an.

SUBEND
kennzeichnet das Ende eines Unterprogramms und übergibt die Programmkontrolle an die dem CALL-Aufruf folgende Anweisung. **S**

SUBEXIT
übergibt die Programmsteuerung vom Ende des Unterprogramms an die dem CALL-Aufruf folgende Programmzeile. **S**

TAB (*numerischer Ausdruck*)
bestimmt die Spaltenposition der nachfolgenden PRINT- oder DISPLAY-Anweisung. **F**

TAN (*numerischer Ausdruck im Bogenmaß*)
berechnet den Tangens des numerischen Ausdrucks. Der Wert des numerischen Ausdrucks wird hierbei als Bogenmaß interpretiert. **F**

TRACE
bewirkt jeweils die Ausgabe der Zeilennummer vor der Ausführung der Anweisung. **C, S**

UNBREAK
hebt alle mit BREAK definierten Stop-Zeilen auf. **C,S**

UNTRACE
hebt den TRACE-Befehl auf. **C, S**

VAL (*Stringausdruck*)
wandelt den Stringausdruck in einen Zahlenwert um. **F**

CALL VCHAR (*Zeile, Spalte, Zeichencode,* [, *Wiederholungszahl*])
setzt das mit dem Zeichencode definierte Zeichen an die mit Zeile und Spalte angegebene Bildschirmposition. Mit der Wiederholungsanzahl wird angegeben, wie oft dieser Vorgang wiederholt wird. **S, C**

CALL VERSION (*numerische Variable*)
ordnet einen der laufenden BASIC-Version entsprechenden Wert einer numerischen Variablen zu. **S, C**

Literaturverzeichnis

Peter Ickenroth: Planen, Kalkulieren, Kontrollieren mit BASIC Taschenrechnern. SYBEX-Verlag GmbH, Düsseldorf 1983

Stanley R. Trost: Programme für meinen Apple II. SYBEX-Verlag GmbH, Düsseldorf 1983

Edmund Heinen: Industriebetriebslehre – Entscheidungen im Industriebetrieb. Gabler Verlag, Wiesbaden 1974

R. Sellien und H. Sellien: Dr. Gabler's Wirtschaftslexikon. Gabler Verlag, Frankfurt 1972

J. R. Tiedke und H.W. Witthof: Betriebswirtschaftslehre der Industrie. Bad Homburg vor der Höhe 1980

TI BASIC / Extended BASIC für Anfänger und Fortgeschrittene. Programmierhandbuch für Texas Instruments Home Computer. Texas Instruments Deutschland GmbH, Learning Center Freising 1983

TI Extended BASIC for the TI-99/4 Home Computer. Texas Instruments Incorporated 1981

Rainer Bernert: TMS 9900. Assembler Handbuch für das Mini Memory. Texas Instruments Ges. m. b. H. Wien 1983

T. Hartnell: Sinclair ZX Spectrum. SYBEX-Verlag GmbH, Düsseldorf 1983

Index

Die SYBEX Bibliothek – eine Auswahl

MEIN ERSTER COMPUTER (2. überarbeitete Ausgabe)
von Rodnay Zaks – eine Einführung für alle, die den Kauf oder die Nutzung eines Mikrocomputers erwägen. 320 Seiten, 150 Abbildungen, Format DIN A5, Ref.Nr.: **3020** (1982)

CP/M HANDBUCH MIT MP/M
von Rodnay Zaks – ein umfassendes Lehr- und Nachschlagewerk für CP/M, das Standard-Betriebssystem für Mikrocomputer. 322 Seiten, 100 Abbildungen, Format DIN A5, Ref.Nr. **3002** (1981)

MIKROPROZESSOR INTERFACE TECHNIKEN (3. überarbeitete Ausgabe)
von Rodnay Zaks/Austin Lesea – Hardware- und Software-Verbindungstechniken samt Digital/Analog-Wandler, Peripheriegeräte, Standard-Busse und Fehlersuchtechniken. 425 Seiten, 400 Abbildungen, Format DIN A5, Ref.Nr.: **3012** (1982)

PROGRAMMIERUNG DES 6502 (2. überarbeitete Ausgabe)
von Rodnay Zaks – Programmierung in Maschinensprache mit dem Mikroprozessor 6502, von den Grundkonzepten bis hin zu fortgeschrittenen Informationsstrukturen. 360 Seiten, 160 Abbildungen, Format DIN A5, Ref.Nr.: **3011** (1982)

POCKET MIKROCOMPUTER LEXIKON
– die schnelle Informations-Börse! 1300 Definitionen, Kurzformeln, technische Daten, Lieferanten-Adressen, ein englisch-deutsches und französisch-deutsches Wörterbuch. 176 Seiten, Format DIN A6, Ref.Nr. **3008** (1982)

BASIC COMPUTER SPIELE/Band 1
herausgegeben von David H. Ahl – die besten Mikrocomputerspiele aus der Zeitschrift „Creative Computing“ in deutscher Fassung mit Probelauf und Programmlisting. 200 Seiten, 56 Abbildungen, Ref.Nr. **3009**

BASIC COMPUTER SPIELE/Band 2
herausgegeben von David H. Ahl – 84 weitere Mikrocomputerspiele aus „Creative Computing“. Alle in Microsoft-BASIC geschrieben mit Listing und Probelauf. 220 Seiten, 61 Abbildungen, Ref.Nr.: **3010**

MEIN ERSTES BASIC PROGRAMM
von Rodnay Zaks – das Buch für Einsteiger! Viele farbige Illustrationen und leichtverständliche Diagramme bringen Spaß am Lernen. In wenigen Stunden schreiben Sie Ihr erstes nützliches Programm. 218 Seiten, illustriert, Ref.-Nr.: **3033** (November 1983)

BASIC PROGRAMME – MATHEMATIK, STATISTIK, INFORMATIK
von Alan Miller – eine Bibliothek von Programmen zu den wichtigsten Problemlösungen mit numerischen Verfahren, alle in BASIC geschrieben, mit Musterlauf und Programmlisting. 352 Seiten, 120 Abbildungen, Ref.Nr.: **3015** (1983)

PLANEN UND ENTSCHEIDEN MIT BASIC
von X. T. Bui – eine Sammlung von interaktiven, kommerziell-orientierten BASIC-Programmen für Management- und Planungsentscheidungen. 200 Seiten, 53 Abbildungen, Ref.-Nr.: **3025** (1983)

BASIC FÜR DEN KAUFMANN
von D. Hergert – das BASIC-Buch für Studenten und Praktiker im kaufmännischen Bereich. Enthält Anwendungsbeispiele für Verkaufs- und Finanzberichte, Grafiken, Abschreibungen u.v.m. ca. 240 Seiten, 76 Abbildungen, Ref.-Nr.: **3026** (November 1983)

PLANEN, KALKULIEREN, KONTROLLIEREN MIT BASIC-TASCHENRECHNERN
von P. Ickenroth – präsentiert eine Reihe von direkt anwendbaren BASIC-Programmen für zahlreiche kaufmännische Berechnungen mit Ihrem BASIC-Taschenrechner. 141 Seiten, 45 Abbildungen, Ref.-Nr.: **3032** (1983)

EINFÜHRUNG IN DIE TEXTVERARBEITUNG
von Hal Glatzer – woraus eine Textverarbeitungsanlage besteht, wie man sie nutzen kann und wozu sie fähig ist. Beispiele verschiedener Anwendungen und Kriterien für den Kauf eines Systems. 246 Seiten, 67 Abbildungen, Ref.Nr. **3018** (1983)

EINFÜHRUNG IN WORDSTAR
von Arthur Naiman – eine klar gegliederte Einführung, die aufzeigt, wie das Textbearbeitungsprogramm WORDSTAR funktioniert, was man damit tun kann und wie es eingesetzt wird. 238 Seiten, 30 Abbildungen, Ref.Nr.: **3019** (1983)

ERFOLG MIT VisiCalc
von D. Hergert – umfassende Einführung in VisiCalc und seine Anwendung. Zeigt Ihnen u. a.: Aufstellung eines Verteilungsbogens, Benutzung von VisiCalc-Formeln, Verwendung der DIF-Datei-Funktion. Ca. 224 Seiten, 58 Abbildungen, Ref.-Nr.: **3030** (Herbst 1983)

6502 ANWENDUNGEN
von Rodnay Zaks – das Eingabe-/Ausgabe-Buch für Ihren 6502-Microprozessor. Stellt die meistgenutzten Programme und die dafür notwendigen Hardware-Komponenten vor. 282 Seiten, 205 Abbildungen, Ref.Nr.: **3014** (1983)

BASIC ÜBUNGEN FÜR DEN APPLE
von J.-P. Lamoitier – das Buch für APPLE-Nutzer, die einen schnellen Zugang zur Programmierung in BASIC suchen. Abgestufte Übungen mit zunehmendem Schwierigkeitsgrad. 252 Seiten, 185 Abbildungen, Ref.Nr.: **3016** (1983)

PROGRAMME FÜR MEINEN APPLE II
von S. R. Trost – enthält eine Reihe von lauffähigen Programmen samt Listing und Beispiellauf. Hilft Ihnen, viele neue Anwendungen für Ihren APPLE II zu entdecken und erfolgreich einzusetzen. 192 Seiten, 192 Abbildungen, Ref.-Nr.: **3029** (1983)

BASIC ÜBUNGEN FÜR DEN IBM PERSONAL COMPUTER
von J.-P. Lamoitier – vermittelt Ihnen BASIC durch praktische und umfassende Übungen anhand von realistischen Programmen: Datenverarbeitung, Statistik, kommerzielle Programme, Spiele u.v.m. 256 Seiten, 192 Abbildungen, Ref.-Nr.: **3023** (1983)

PROGRAMMSAMMLUNG ZUM IBM PERSONAL COMPUTER
von S. R. Trost – mehr als 65 getestete, direkt einzugebende Anwenderprogramme, die eine weite Palette von kaufmännischen, persönlichen und schulischen Anwendungen abdecken. 192 Seiten, 158 Abbildungen, Ref.-Nr.: **3024** (1983)

CHIP UND SYSTEM: Einführung in die Mikroprozessoren-Technik
von Rodnay Zaks – eine sehr gut lesbare Einführung in die faszinierende Welt der Computer, vom Microprozessor bis hin zum vollständigen System. Ca. 560 Seiten, 325 Abbildungen, Ref.Nr.: **3017** (Erscheint Januar 1984)

VORSICHT! Computer brauchen Pflege
von Rodnay Zaks – das Buch, das Ihnen die Handhabung eines Computersystems erklärt – vor allem, was Sie damit nicht machen sollten. Allgemeingültige Regeln für die pflegliche Behandlung Ihres Systems. 240 Seiten, 96 Abbildungen, Ref.Nr.: **3013** (1983)

MEIN SINCLAIR ZX81
von D. Hergert – eine gut lesbare Einführung in diesen Einplatincomputer und dessen Programmierung in BASIC. 173 Seiten, 45 Abbildungen, Ref: **3021** (1983)

SINCLAIR ZX81 BASIC HANDBUCH
von D. Hergert – vermittelt Ihnen das vollständige BASIC-Vokabular anhand von praktischen Beispielen, macht Sie zum Programmierer Ihres ZX81. 181 Seiten, 120 Abbildungen, Ref.-Nr.: **3028** (1983)

SINCLAIR ZX SPECTRUM Programme zum Lernen und Spielen
von T. Hartnell – ein Buch zur praktischen Anwendung. Grundzüge des Programmierens aus dem kaufmännischen Bereich sowie Spiele, Lehr- und Lernprogramme in BASIC. 224 Seiten, 120 Abbildungen, Ref. Nr. **3022** (1983)

SINCLAIR ZX SPECTRUM BASIC HANDBUCH
von D. Hergert – eine wichtige Hilfe für jeden SPECTRUM-Anwender. Gibt eine Übersicht aller BASIC-Begriffe, die auf diesem Rechner verwendet werden können, und erläutert sie ausführlich anhand von Beispielen. Ca. 224 Seiten, ca. 150 Abbildungen, Ref.-Nr.: **3027** (Dezember 1983)

FORDERN SIE EIN GESAMTVERZEICHNIS UNSERER VERLAGSPRODUKTION AN:



SYBEX-VERLAG GmbH
Vogelsanger Weg 111
4000 Düsseldorf 30
Tel.: (02 11) 62 64 41
Telex: 8 588 163

SYBEX
6–8, Impasse du Curé
75018 Paris
Tel.: 1/203-95-95
Telex: 211.801 f

SYBEX INC.
2344 Sixth Street
Berkeley, CA 94710, USA
Tel.: (415) 848-8233
Telex: 336311

spielen, lernen, arbeiten mit dem ti-99/4A

Entdecken Sie die faszinierende Welt der Mikrocomputer – die Vorteile sind auf Ihrer Seite!

Dieses Buch führt Sie schrittweise in den richtigen Umgang mit Ihrem TI99/4A. Anhand von vielen Beispielprogrammen lernen Sie, wie Sie sich die Fähigkeiten Ihres Computers beim Einsatz für Arbeit und Spiel wirksam zunutze machen können. Eine eingehende Erklärung der Bedienung Ihres Rechners und eine Einführung in die Programmierung Ihres TI99/4A, in Basic, lassen Sie schnell zum fortgeschrittenen Anwender werden.

Mit diesem Buch holen Sie von Anfang an das Beste aus Ihrem TI99/4A heraus!

ISBN 3-88745-039-6

